SUPPORTING TOP-K ITEM EXCHANGE RECOMMENDATIONS IN LARGE ONLINE COMMUNITIES

SU ZHAN

Bachelor of Engineering

Fudan University, China

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2012

ACKNOWLEDGEMENT

I would like to thank all people who have supported, helped and inspired me during my study.

I especially wish to express my deep and sincere gratitude to my supervisor, Professor Anthony K.H. Tung for his patience, motivation, enthusiasm, and immense knowledge. His invaluable support helped me in all time of my research and thesis writing. His conscientious attitude of working set me a good example.

I warmly thank Dr. Zhenjie Zhang for his valuable advice and friendly help. He introduced me to the item exchange problem and we worked together on it. He gave me important guidances on problem solving and paper writing and kindly improved my paper.

I wish to thank all my lab-mates in the Database Lab. These people with the good intellegence and friendship make our lab a convivial place for working. During my four years in the lab, we worked and played together. They inspired me in both research and life.

I would like to thank my girlfriend Zhou Yuan for her encouragement and understanding during my study.

Last but not least, thank my parents for their endless love and support.

CONTENTS

SUMMARY

Item exchange is becoming popular in many online community systems, e.g. online games and social network web sites. Traditional manual search for possible exchange pairs is neither efficient nor effective. Automatic exchange pairing is increasingly important in such community systems, and can potentially lead to new business opportunities. To facilitate item exchange, each user in the system is entitled to list some items he/she no longer needs, as well as some required items he/she is seeking for. Given the values of all items, an exchange between two users is eligible if 1) they both have some unneeded items the other one wants, and 2) the exchange items from both sides are approximately of the same total value. To efficiently support exchange recommendation with frequent updates on the listed items, new data structures are proposed in this thesis to maintain promising exchange pairs for each user. Extensive experiments on both synthetic and real data sets are conducted to evaluate our proposed solutions.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Item exchange is becoming popular and widely supported in more and more online community systems, e.g. online games and social network web sites. For example, *Frontier Ville*, one of the most popular farming games with millions of players, every individual player only owns limited types of resources. To finish the tasks in the game, the players can only resort to their online neighborhood for resource exchanges [1]. Due to the lack of effective channel, most of the players are now relying on the online forum to look for the exchange poster., posting the unneeded and wishing items to attract other users meeting the exchange requirements. While the items for exchange in online games are usually virtual objects, there are also some emerging web sites dedicated to the exchange services on second-hand commodities. *Shede* [6], for example, is a quick-growing internet-based product exchange platform in China, reaching millions of transactions every year. Similar web sites have also emerged in other countries, e.g. UK [5], Singapore [2] et al. However, the users on the platform are only able to find matching exchange parties by browsing or searching with keywords in the system. Despite the huge potential value of the exchange market, there remains a huge gap between the increasing demands and the techniques supporting automatic exchange pairing.

In this thesis, we aim to bridge this gap with an effective and efficient mechanism to support automatic exchange recommendations in large online communities. Generally speaking, a group of candidate exchanges are maintained and displayed to each user in the system, suggesting the most beneficial exchanges to them. The problem of online exchange recommendation is challenging for two reasons. First, it is important to design a reasonable and effective exchange model, on which all users in the system are willing to follow. Second, a system, which can keep user updated with the most recent and acceptable exchange options and handle massive real-time updates, is needed.

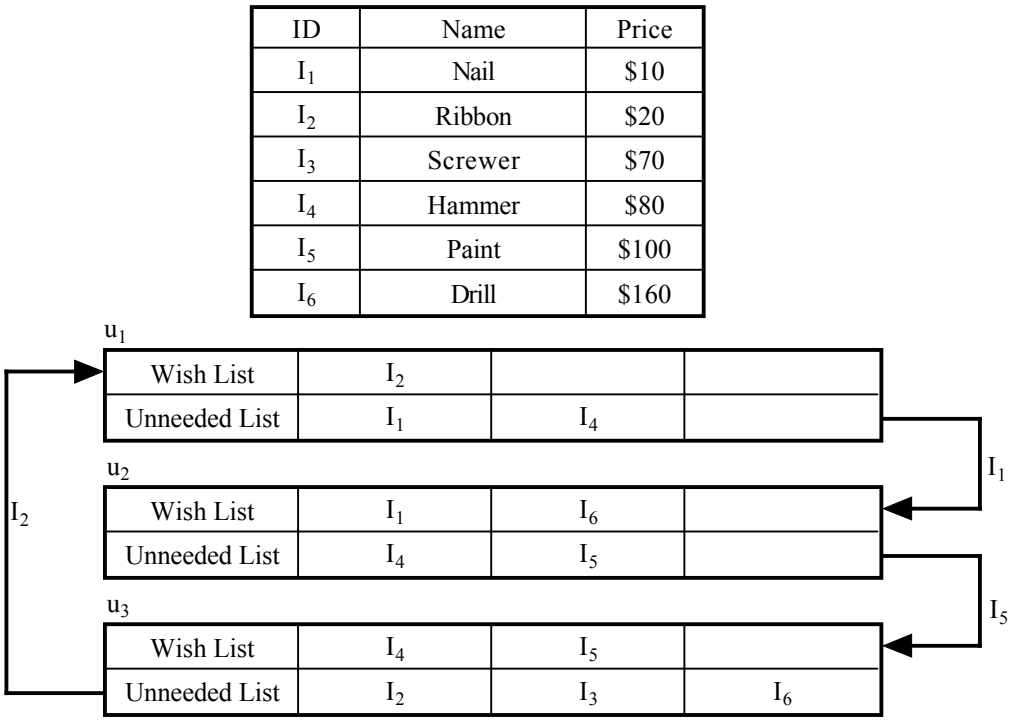


Figure 1.1: Example of transaction in CSEM

To model the behaviors and requirements of the users in the community system [21], some online exchange models have been proposed. The recent study in [7], for example, proposed a Circular Single-item Exchange Model (CSEM). Specifically, given the users in the community, an exchange ring is eligible if there is a circle of

users $\{u_1 \to u_2 \to \ldots u_m \to u_1\}$ that each user u_i in the ring receives a required item from the previous user and gives an unneeded item to the successive user. Despite of the successes of CSEM in kidney exchange problem [10], this model is not applicable in online community systems for two reasons. First, CSEM does not consider the values of the items. The exchange becomes unacceptable to some of the users in the transaction, if he/she is asked to give up valuable items and only gets some cheap items in return. Second, single-item constraint between any consecutive users in the circle limits efficiencies of online exchanges. Due to the complicated protocol of CSEM, each transaction is committed only after all involved parties agree with the deal. The expected waiting time for each transaction is unacceptably long in online communities. In Figure 1.1, we present an example to illustrate the drawbacks of CSEM. In this example, there are three users in the system, $\{u_1, u_2, u_3\}$, whose wishing items and unneeded items are listed in the the rows respectively. Based on the protocol of CSEM, one plausible exchange is a three-user circle, I_1 from u_1 to u_2, I_2 from u_3 to u_1 and I_5 from u_2 to u_3, as is shown with the arrows in Figure 1.1. This transaction is not satisfactory with u_2, since I_5 is worth 100\$ while I_1's price is only 10\$.

In this thesis, we present a new exchange model, called Binary Value-based Exchange Model (BVEM). In BVEM, each exchange is run between two users in the community. An exchange is eligible, if and only if the exchanged items from both sides are approximately of the same total value. Recall the example in Figure 1.1, a better exchange option between u_2 and u_3 is thus shown in Figure 1.2. In this transaction, u_2 gives two items I_4 and I_5 at total value at \$180, while u_3 gives a single item I_6 at value 170\$. The difference between the exchange pair is only 10\$, or 5.9% of the counterpart. This turns out to be a fair and reasonable deal for both users. On the other hand, each exchange in BVEM only involves two users, which

greatly simplifies the exchange procedure. Both of these features make BVEM a practical model for online exchange, especially in highly competitive environment such as online games. To improve the flexibility and usefulness of BVEM model for online communities, we propose a new type of query, called *Top-K Exchange Recommendation*. Upon the updates on the users' item lists, the system maintains the top valued candidate exchange pairs for each user to recommend promising exchange opportunities.

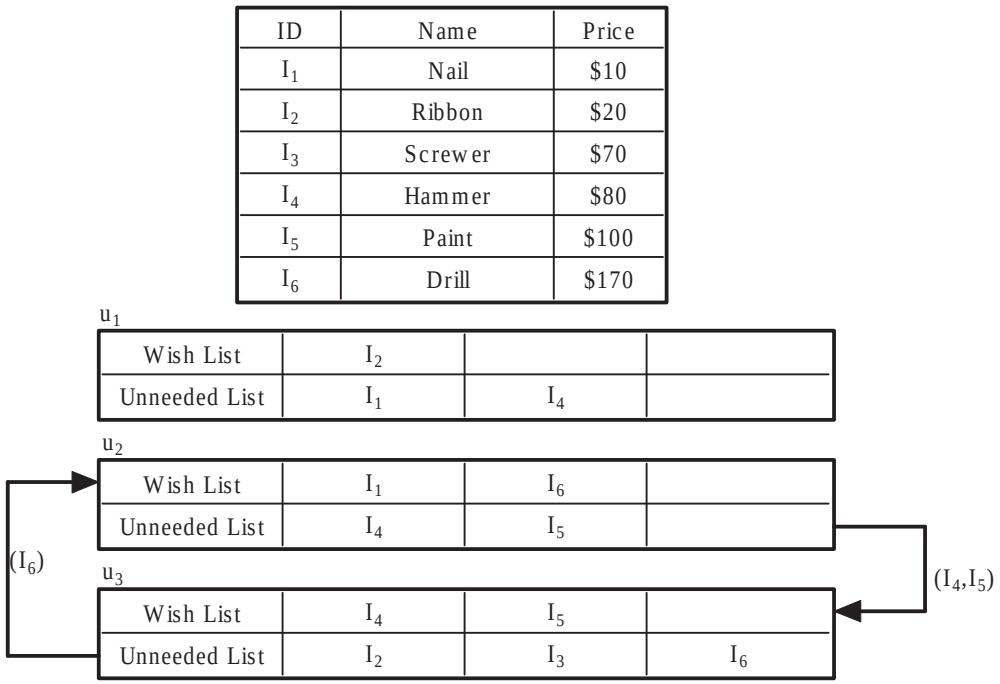


Figure 1.2: Example of transaction in BVEM

Despite the enticing advantages of top-k exchange query under BVEM in terms of effectiveness, scalability becomes an issue. Later in Chapter 3, we prove that given a pair of users in the community, the problem of finding matching exchange pair with the highest total value is NP-hard, with exponential complexity in term of the number of items a user owns(Theorem 3.1). Fortunately, the size of the item lists are usually bounded by some constant number in most of the community systems, leading to acceptable computation cost on the search for the best exchange

plan between two specified users. The problem tends to be more complicated if the community system is highly dynamic, with frequent insertions and deletions on the item lists of the users. To overcome these challenges on the implementation of BVEM, we propose a new data structure to index the top-k optimal exchange pairs for each user. Efficient updates on both insertions and deletions are well supported by our data structure, to maintain the candidate top-k exchange pairs.

We summarize the contributions of the thesis as listed below:

1. We propose the Binary Value-based Exchange Model, capturing the requirements of online exchange behavior.

2. We design a new data structure for effective and efficient indexing on the possible exchange pairs among the users.

3. We apply optimization techniques to improve the efficiency of the proposed index structure.

4. We present extensive experimental results to prove the usefulness of our proposals.

The remainder of the thesis is organized as follows. Chapter 2 reviews some related work on online exchange models and methods. Chapter 3 presents the problem definition and preliminary knowledge of our problem. Chapter 4 discusses the solution to the Top-K Exchange Pair Monitoring Problem. Chapter 5 evaluates our proposed solutions with synthetic data sets and Chapter 6 concludes this thesis.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we survey related work from various areas. We first review exchange and allocation models studied by computational economic communities and database communities. Then we summarize existing research work in recommender systems.

2.1 Related Exchange and Allocation Models

The exchange behaviour has attracted attention of both economists and computer scientists. Economics researchers have proposed various economic models of the matching, allocation and exchange of indivisible goods and resource (e.g. jobs, houses and etc.)[51]. These economic models are mathematical representation of a certain type of exchange activity. Based on these models, mathematical analysis and computer simulation can be done to reveal the characteristics of the activity (e.g. if a equilibrium state exists in the exchange market). On another hand, computer science researchers have also study the exchange model[8, 17]. They are interested in efficiently finding centralized exchange arrangement by computer simulation. Moreover, they develop exchange recommender system in large community network based on their proposed exchange models.

In the following subsections, we review several exchange models, including house allocation and exchange models, kidney exchange models and the circular single-item exchange model.

2.1.1 House Allocation and Exchange

In this subsection, we introduce two highly related problems about the house allocation and exchange: the house allocation problem and the housing market problem.

House Allocation

The **house allocation problem** is first introduced in [27], in which a preference-based allocation model is proposed and applied to the assignment of freshmen to upper-class houses (residence halls) at Harvard College. Following [51], the house allocation problem is defined as:

Definition 2.1. ***House Allocation Problem*** *Given A, H and $\succ$.*
$A = \{a_1, a_2, \ldots, a_n\}$, referring to n agents who want to buy houses.
$H = \{h_1, h_2, \ldots, h_n\}$ is n houses for sale.
$\succ = \{\succ_a \mid a \in A\}$, and each $\succ_a$ is a strict order relation, indicating a's preference over houses. $h_i \succ_a h_j$ means a prefer house h_i rather than h_j.

*Output a **matching**, which is a bijection $\mu : A \mapsto H$. $\mu(a)$ is the house assigned to a.*

Although the problem is defined on house allocation, it can also be generalized to allocation of indivisible resource/goods.

Let $\phi(A, H, \succ)$ denotes the house allocation mechanism (algorithm), which takes A, H and $\succ$ as input and a matching as the output. When A and H are fixed, for simplicity we use $\phi(\succ)$ to indicate the algorithm.

A matching μ is **Pareto-efficient**, if there exists no other matching μ', such that for all $a \in A$, $\mu(a) \not\succ_a \mu'(a)$ and for some $a \in A$, $\mu'(a) \succ_a$. Namely, in a Pareto-efficient matching, no agent can be re-assigned a more preferable house without other agents being made worse off. A house allocation algorithm $\phi(A, H, \succ)$ is Pareto-efficient, if for any input, it always outputs a Pareto-efficient matching.

An algorithm $\phi(\succ)$ is **Strategy-proof**, if for all agent a, there exists no $\succ_a^*$, such that $\mu(\succ \setminus \succ_a \cup \succ_a^*) \succ_a \phi(\succ)$. That is, an agent can never be benefitted by telling their preference strategically rather than faithfully.

A family of mechanisms called **serial dictatorships**[48] solves the allocation problems in a dictatorial manner. In these mechanisms, a priority ranking, which is a bijection $f : \{1, 2, \ldots, n\} \mapsto \{1, 2, \ldots, n\}$, is assigned to all agents. Agents are allocated houses one-by-one in the ascending order of $f(a)$. Each agent is assigned with her/his most preferable house among the remaining houses that are not assigned to a higher ranked agent. Algorithm 1 formally describe the mechanism.

Algorithm 1 Serial Dicatatorship$\mu(A, H, \succ, f)$

```
sort A in ascending order of f(a).
for a ∈ A do
    assign a with her top choice h in H.
    remove h from H.
end for
```

In [9], it is proven that *A matching mechanism is Pareto-efficient if and only if it is a serial dictatorship*, which means: 1) serial dictatorship mechanism is Pareto-efficient and 2) for any Pareto-efficient matching μ, there is a priority ranking f that induces the matching μ.

Housing Market

Next we consider a second model, which is an exchange-based model, called the **housing market**[49]. This model differs from the house allocation in only one

aspect: each house is initially owned by an agent. This ownership is called **(initial) endowment**. Formally, the house market is defined as follow:

Definition 2.2. ***Housing Market Problem*** *Given A, H, h and $\succ$.*
$A = \{a_1, a_2, \ldots, a_n\}$, referring to n agents who want to buy houses.
H is a set of n houses in the market.
$h : A \mapsto H$ is a bijection between agents and houses. h_a denote the house initially owned by agent a.
$\succ = \{\succ_a | a \in A\}$, and each $\succ_a$ is a strict order relation, indicating a's preference over houses. $h_i \succ_a h_j$ means that a prefers house h_i rather than h_j.

Output a matching, which is the same as the output of house allocation problem..

Unlike the house allocation, in which a central planner arranges the allocation, a housing market is an exchange market, where decentralized trading among agents are done. Agents have the right to refuse a exchange proposal without benefit. Therefore, **individually rational** is introduced. A matching μ is individually rational, if for all agents $a \in A, h_a \not\succ_a \mu(a)$. That is, no agent trades her house for a less preferable one. A mechanism is individually rational if it always outputs an individually rational matching for each input.

A second concept that we introduce is **competitive equilibrium**. Let the price vector be $p = \{p_h | h \in H\}$, where p_h is the price of the house h. A competitive equilibrium is a matching-price vector pair (μ, p), subject to:

- **Budget Constraint** $p_{\mu(a)} \leq p_{h_a}$.
- **Utility Maximization** $\forall h \in H$, if $p_h \leq p_{h_a}$, $h \not\succ_a \mu(a)$.

The competitive equilibrium is a balanced market state, in which each agent owns the most preferred house that she can afford. However, it is not immediately clear if

the competitive equilibrium exists for all housing markets. The theoretical analysis of competitive equilibrium relies on another important concept called the core.

We say a coalition (subset) of agents $B \subseteq A$ **blocks** a matching μ, if there exists another matching ν, such that:

- $\forall a \in B, \exists b \in B$ such that $\nu(a) = h_b$,
- $\forall a \in B, \mu(a) \not\succ_a \nu(a)$,
- $\exists a \in B, \nu(a) \succ_a \mu(a)$.

In another word, a matching is blocked by a group of agents, if these agents benefit from excluding other agents and only trading within the group. A matching μ is in the **core**, if and only if it is blocked by no coalitions of agents. The core is a stable market state. However, it is not apparent that the core exists. In [49], the following theorem is proven, which shows the existence of the core and competitive equilibrium in housing markets, and also reveals the connection between them.

Theorem 2.1. *The core of a housing market is non-empty and there exists a core matching that can be sustained as part of a competitive equilibrium.*

In [49], a constructive method is used to prove the theorem. The authors propose the David Gale's **Top Trading Cycle** algorithm, which finds the core matching. It is illustrated in Algorithm 2.

In each iteration of the while-loop, a graph G is constructed. Its vertices correspond to agents and houses. In G, each agent points to her most preferable house and each house points to its initial owner. It is readily to prove that G contains at least one cycle, and all cycles in G are non-intersecting. Therefore, we can safely assign each agent in the cycle with her top choice, which is the node she points to in the cycle. After removing these agents and their houses, the algorithm enters a new iteration. It terminates until all agents are assigned a house.

Algorithm 2 Gale's Top Trading Cycle$(A, H, h, \succ)$

1: **while** $A \neq \emptyset$ **do**
2: Construct an empty directed graph $G = (V, E)$.
3: Set $V = A \cup H$
4: For each $a \in A$, $E = E \cup \{(h_a, a)\}$
5: For each $a \in A$, let h_a^* be a's current top choice, $E = E \cup \{(a, h_a^*)\}$
6: If $a \in A$ is in any graphic cycle, assign h_a^* to it.
7: If an agent a is assigned a house, remove a from A and remove h_a from H.
8: **end while**

Theorem 2.2. *Output of Gale's Top Trading Cycle is a core matching, and is also sustainable by a competitive equilibrium.*

A competitive equilibrium price vector can be constructed as follow: 1) all houses that are removed in a same iteration in Algorithm 2 are assigned with a same price; 2) all houses that are removed in later iterations are assigned a price lower than the current house. That is, the later a house is removed in the Algorithm 2, the lower its price is.

In [41], it is proven that if no agents is indifferent between any houses ($\succ_a$ is a strict preference for any $a \in A$), the core is always non-empty, contains exactly one matching and is the unique matching that can be sustained at a competitive equilibrium.

In [40], the core mechanism is also proven to be strategy-proof.

It is easy to see that the core mechanism also has several positive properties: individually rational and Pareto-efficient and strategy-proof. In [32], a stronger theorem shows that it is a dominating mechanism. That is, a mechanism is individually rational, Pareto-efficient and strategy-proof for a housing market only if it is core mechanism.

However, these good properties may not hold for a more complex model. In [28], authors study a model in which there are Q types of goods(house). Each agent owns exactly one good of each type. Exchange can be done only among the same

type of goods. Each agent has a strict utility score for each good. The overall utility score of a Q-good combination is the sum of all Q utility scores. Agents pursuit high utility by exchanging goods. In their economy model, the core maybe empty. Moreover, the competitive equilibrium matching is proven to be in the core, but a core is not sufficiently sustained at a competitive equilibrium. That is, the set of competitive equilibrium matchings can be smaller than the core. In addition, there is no mechanism that is individually rational, Pareto-efficient and strategy-proof.

2.1.2 Kidney Exchange

In this subsection, we consider an important application of exchange models, which is the kidney exchange. Kidney exchange is a project aiming to improve the probability that a patient waiting for kidney transplanting finds a compatible donor and shorten their waiting time. To adapt to the restriction imposed by the nature of the problem, new models are developed and a new theory is constructed.

Background

Kidney transplanting is the organ transplant of a kidney into a patient with end-stage renal disease. Since organ trading is illegal in many countries, donation is the only source of kidneys in these countries. Depending on the source, the donation can be classified as living donor and deceased donor. The living donors are generally friends or relatives of a patient. These donors are only willing to donate one of their kidneys to a designated patient.[1] Deceased donor are assigned according to a centralized priority mechanism in the US and Europe[51]. Based on the mechanism, the patients are ordered in a waiting list. A donor kidney is assigned to a selected patient based on a metric considering the degree of match,

[1]There exist Good Samaritan donors who donate their kidneys to strangers. However, the number of these donors is small relative to the number of directed live donors[51].

waiting time and other medical and fairness criteria.

However, the living donor may not be compatible with a patient. The compatibility test is conducted for each donor and patient. There are two kinds of compatibility tests:

- **Blood compatibility test**. This test verifies if the donor's blood is compatible with patient's blood. For example, in the ABO blood group, "A" blood-type donor is blood-type compatible with "A" and "AB" blood type patient.

- **Tissue compatibility test** (or **crossmatch test**). This test examines the human leukocyte antigen (HLA) in patient's and donor's DNA. The patient and the donor are tissue type incompatible if the patient's blood contains antibodies against donor's human leukocyte antigen (HLA) proteins.

Traditionally, incompatible donors are sent home. To better utilize them, kidney exchange is applied. There are two ways of kidney exchange:

- **List exchange** List exchange allows exchange between an incompatible patient-donor pair and the deceased donor waiting list. The donor's kidney can be assigned to another compatible patient in the waiting list. In return, the patient becomes the first priority person in the waiting list.

- **Paired exchange** Paired exchange can be applied among multiple incompatible patients-donor pairs. In paired exchange, a patient receives a transplant from the donor of another pair, and his paired donor donates the kidney to feasible patient of other pairs.

Moreover, besides medical compatibility which is crucial, the preference of patients and doctors are also important. Based on several factors, such as geographic

distance of the match, patients and doctors have a preference over the compatible donors or even refuse exchange with some donors. This should also be considered in the model.

Kidney exchange programs have been established in several countries, such as the USA[3], the UK[4] and Romania[31].

Exchange Model

The general kidney exchange model is defined as follow:

Definition 2.3. ***Kidney Exchange Model*** *A kidney exchange model consists of:*

- *a set of patients* $P = \{p_1, \ldots, p_n\}$.
- *a set of donors* $D = \{d_1, \ldots, d_n\}$.
- *a set of donor-patient pairs* $\{(d_1, p_1), \ldots, (d_n, p_n))\}$
- *a set of compatible donors* $\{D_1, \ldots, D_n\}$*, where* $D_i \subseteq D$*, indicating the donors compatible with patient* p_i.
- *a set of strict preference relations* $\succ = \{\succ_1, \ldots, \succ_n\}$. $succ_i$ *is an ordered relation over* $D_i \cup \{w\}$*, denoting* p_i*'s preference over her compatible donors.* w *refers to the patient's option to become the priority person in the deceased waiting list in return of exchange her paired donor.*

The output of the kidney exchange problem is a matching between $D \cup \{w\}$ and P, indicating the assignment of donors or waiting list option to every patient.

A matching μ is **Pareto-efficient** if there is no other matching η such that all patients are assigned a donor in η no worse than in μ, and some patients are assigned a donor in η better than in μ. A mechanism is Pareto-efficient if it always output Pareto-efficient matching.

A matching is **individually rational** if for each patient, the matched donor is not worse than her paired-donor. A mechanism is individually rational if it always selects an individually rational matching.

A mechanism is **strategy-proof** if no agent can be better off by strategically rather truthfully reporting their preference and paired-donors.

In the remaining part of this subsection, we review the recent work on kidney exchange models, including the general model with strict preference and its variants with extra assumptions/restrictions.

Multi-way Kidney Exchanges with Strict Preference

In [43], the multi-way kidney exchange problem is studied. It follows the definition 2.3, which means:

- List exchanges are allowed.
- Paired exchanges are allowed. The exchange cycle can be of any length.
- Each patient has a strict preference over the donors. That is, no two donors are equally preferable to a patient.

In [43], the **top trading cycles and chains(TTCC) algorithm** is proposed to solve the problem. Similar to Gale's top trading cycles algorithm, this algorithm construct a directed graph from the input following the steps:

- create a vertex for each patient, each donor and the waiting list option w.
- add an edge from each patient's donor to the patient.
- add an edge from each patient to her most preferable kidney. If no compatible kidney is there, point the patient to w.

In this graph, a w-chain is defined as a path starting with a donor and end with the w. It is easy to prove that there exists at least a w-chain if no cycle exists. Based on this, TTCC works as shown in Algorithm 3. In each iteration, it finds a w-chain or a cycle and removes it. In line 8, a chain selection rules is used. It determines which w-chain to choose. Moreover, in line 11, it also determines if the "tail donor", which is the donor staring the w-chain, should be removed or kept for the remaining iteration. If the tail donor is removed, it is finally assigned to the deceased waiting list and not participates in the paired exchange. Depending on different chain selection rules, TTCC outputs different matchings. We list a few candidate rules below:

Algorithm 3 TTCC algorithm

```
1:  while not all patients are assigned a donor/waiting list do
2:      Construct a graph G based on current patients and donors.
3:      while there exists a graphic cycle in G do
4:          assign each patients in the cycle with the donor that she points to.
5:          remove the patients and donors in the cycle.
6:      end while
7:      if there exits a w-chain then
8:          select a w-chain according to a chain selection rule.
9:          assign each patient in the w-chain with the donor/waiting list that she
            points to.
10:         remove the patients and donors in the w-chain (do not remove w).
11:         according to the chain selection rule, either remove the "tail donor" or
            keep it.
12:     end if
13: end while
```

1. Select the minimal w-chain and remove the tail donor.

2. Select the longest w-chain and remove the tail donor.

3. Select the longest w-chain and keep the tail donor.

4. Assign a priority ranking to the patient-donor pairs (as in the serail dictatorships). Select the w-chain starting with the highest ranked pair and remove the tail donor.

5. Assign a priority ranking to the patient-donor pairs. Select the w-chain staring with the highest ranked pair and keep the tail donor.

In [43], authors show that different rules result in different characteristics:

Theorem 2.3. *If the w-chain selection rules keep the tail donor, the induced TTCC algorithm is Pareto-efficient.*

Theorem 2.4. *The TTCC algorithm induced by rule 1, 4 or 5 is strategy-proof. The TTCC algorithm induced by rule 2 or 3 is not strategy-proof.*

Two-Way Paired Exchanges with 0-1 Preferences

Pervious we consider the kidney exchange with unlimited cycle/chain length. However, it is suggested that the pairwise exchange with 0-1 preferences is a more practicable solution[42]. That is, each exchange involves only two patient-donor pairs, and the patients and doctors are indifferent among compatible donors. This is because 1) all transplantations in an exchange must be carried out simultaneously, in case that a donor would back out after her paired-patient receives a transplantation, and 2) in the United States, transplants of compatible live kidneys have about equal graft survival probabilities regardless of the closeness of tissue types between the patient and the donor[26].

Based on this, we can simplify the exchange model:

Definition 2.4. ***Two-Way Kidney Exchanges Problem*** *Given (P, R):*

- *A set of patient-donor pairs $P = \{p_1, \ldots, p_n\}$.*[2]

[2]In the remaining of this section, we may also use p_i to refer to a patient in the pair if no ambiguity is created.

- *A **mutually compatible** relation $R \subseteq P \times P$. $(p_i, p_j) \in R$ if and only if p_i's patient is compatible with p_j's donor and vice versa.*

Output a matching $M \subseteq R$, such that no patient-donor pair appear in M more than once.

For a given input, we define $\mathcal{M}$ as the set of all feasible matchings. For the sake of fairness, we are interested in the stochastic output of this problem. A **lottery** λ is defined as a probability distribution over all feasible matchings $\lambda = (\lambda_\mu)_{\mu \in \mathcal{M}}$. The **utility** of a patient p_i under a lottery λ is the probability that the patient gets a transplant. It is denoted as $u_i(\lambda)$. The **utility profile** of a lottery λ is $u(\lambda) = \{u_1(\lambda), \ldots, u_n(\lambda)\}$.

Lottery often assigns inequable probability to patients, which is unfair to some patients. We say a utility profile $u(\lambda)$ is **Lorenz-dominant** if for any $k \in \{1, 2, \ldots, n\}$, the sum of utilities of the k most unfortunate (i.e. lowest utility) patients is highest among all feasible utility profile of any lotteries. Lorenz-dominance identify the utility profile has the least possible inequality of the utility.

A matching is **Pareto-efficient** if there is no other matching that makes some patients strictly better off without making any patient worse off. A lottery is **ex-post efficient** if and only if it only assigns non-zero probability to the Pareto-efficient matching. A lottery is **ex-ante efficient** if there is no other lottery that makes some patients strictly better off (i.e. higher utility) without making any patient worse off.

In [42], two lemmas are proven:

Lemma 2.1. *The same number of patients are matched in each Pareto-efficient matching. The number is also maximum among all matchings.*

Lemma 2.2. *A lottery is ex-ante efficient if and only if it is ex-post efficient.*

The first lemma reveals that finding Pareto-efficient matching is equivalent to finding the maximum matching in the graph theory. The second lemma shows that ex-ante efficiency is equivalent to ex-post efficient for the two-way kidney exchanges problem.

In [42], a deterministic algorithm and a lottery algorithm is proposed. The deterministic algorithm achieves the Pareto-efficiency and strategy-proofness. The lottery algorithm is Pareto-efficient and strategy-proof, and its utility profile is always Lorenz-dominant.

Multi-way Paired Exchange with Non-strict Preference

As mentioned earlier, paired exchange with 0-1 preference and short exchange cycle is more practicable. However, it is clear that allowing longer exchange cycle can potentially find paired exchange for more patients. In [44], the authors examined the size of the multi-way exchange in order to find out what has been lost in two-way paired exchange. In their paper, they consider the 2-, 3- and 4-way paired exchange with 0-1 preference. In addition, there are three assumptions:

- **Upper Bound Assumption**. No patients are tissue type incompatible. Only ABO blood type compatibility is considered.
- **Large Population of Incompatible Patient-Donor Pairs**. Let X-Y pair denotes a patient with blood type X and a donor with a blood type Y. We assume that there is an arbitrary many number of O-A, O-B, O-AB, A-AB and B-AB type pairs.
- There is no A-A pair or there are at least two of them. The same is also true for each of the types B-B, AB-AB and O-O.

Base on these assumptions, they solve the theoretical upper bounds of the number of patients that are covered by 2-, 2&3-, 2&3&4-way paired exchanges respectively. Moreover, the following theorem shows that allowing cycle length longer than four is not necessary under their assumptions:

Theorem 2.5. *Consider a kidney exchange problem for which the assumptions above hold and let μ be any maximum matching without restriction on the exchange cycle length. Then there exists a maximal matching ν that consists only of two-, three- and four-way exchanges, under which the same set of patients benefits from exchange as in matching μ.*

In [50], authors synthesize the kidney exchange data based on national recipient characteristics with considering both blood-type and tissue-type compatibility. They compare their simulation results with the theoretical upper bounds in [44]. The result shows that although the upper bounds are developed with ignoring the tissue-type compatibility, it is still predictive. Moreover, two-, three- and four-way exchanges virtually achieve all the possible gains from unrestricted exchanges when the population size is large. This verifies the Theorem 2.5.

Hardness of Finding Multi-way Kidney Exchange

There are other research been done on the exchange algorithm analysis. The TTCC algorithm does not take cycle length into consideration. In [17], a modified exchange model is proposed to overcome this problem. It differs from Definition 2.3 in the below respects:

- No deceased list exchange is allowed. Only paired exchanges are considered. Therefore, the result matching can be described by a permutation π, $\pi(i)$ indicating the donor $d_{\pi(i)}$ is assigned to p_i.

- Patients' actual preference is based on the (donor, cycle length) pairs. The cycle length is the length of the exchange cycle that the patient attends in the current permutation. A patient p_i prefers (d_j, N) than (d_k, M) if:
  - $d_j \succ_i d_k$, or
  - $d_j \sim_i d_k$ and $N < M$.

  That is, the patient prefers a smaller cycle if the donors are indifferent.

Like the housing market, a coalition (subset) of patients **block** a matching μ if all of them can be made weakly better off and some of them can be made strictly better off by only exchange with each other. The **core** of the kidney exchange is the set of matchings that are not blocked by any coalition. A patient p_i is said to be **covered** by a matching μ if $\mu(p_i) \neq d_i$ (i.e. she receives a compatible donor).

It is interesting that if we can find a core matching that covers as many as possible patients. In [17], authors define the deterministic problem below:

Definition 2.5. ***MAX-COVER-KE*** *For a kidney exchange problem, determine if a matching μ covers the maximum number of patients.*

They prove the problem is not only NP-complete, but also inapproximable.

Theorem 2.6. *MAX-COVER-KE is not approximable within $n^{1-\epsilon}$ for any $\epsilon > 0$ unless P=NP.*

In [20], the authors study the cycle length of a core matching. They are interested in the problems that if the cycle length can be shorten. In a matching μ, they define $C_\mu(p_i)$ as the length of the exchange cycle that p_i take part in. If p_i fails to get a compatible donor in μ, $C_\mu(p_i) = +\infty$.

We can easily adapt the top trading cycle algorithm (Algortihm 2) to paired kidney exchange with strict preference (but cycle length is not considered). That

is, construct a graph with patients and donors being the vertices; let each patient point to her top choice donor (points to her paired donor if there is no compatible donor) and each donor points to her paired patient. Then cycles are iteratively removed from the graph and exchange cycles are formed.

In [20], the following problems are proven to be NP-Complete:

- **ALL-SHORTER-CYCLE-KE** For a kidney exchange problem with strict preference, determining if there is a matching μ in the core, such that $C_\mu(p_i) < C_{TTC}(p_i)$ for all $p_i \in P$. Here TTC denotes the matching given by top trading cycle algorithm.

- **3-CYCLE-KE** For a kidney exchange problem with strict preference, determining if there is a matching μ in the core, such that $C_\mu(p_i) \leq 3$ for all $p_i \in P$.

- **FULL-COVER-KE** For a kidney exchange problem with strict preference, determining if there is a matching μ in the core, such that $\mu(p_i) \neq d_i$ (i.e. the patient is assigned a compatible donor) for all $p_i \in P$.

2.1.3 Circular Single-item Exchange Model

In this subsection, we introduce the Circular Single-item Exchange Model (CSEM), which is closely related to the kidney exchange model that we introduce in the last subsection. This model is proposed in [8]. In this subsection, all stated results are from this paper unless otherwise noted.

This model is based on a real-life application, which is also the main problem of this thesis: users want to trade their unneeded goods for what they want in an online social network. There are two CSEM models, a deterministic model called

simple exchange markets and its probabilistic version is called **probabilistic exchange markets**.

The simple exchange markets assume that each user has two lists: an **item list** and a **wish list**. The item list contains all her unneeded items, which are ready to be given away. The wish list contains all her wanted items, which are the items that she needs. The formal definition is given below:

Definition 2.6. ***Simple Exchange Markets*** *The simple exchange market is a tuple* (U, I, S, W).

- $U = \{u_1, \ldots, u_n\}$ *is the set of users in the market.*
- $I = \{i_1, \ldots, i_m\}$ *is the set of items in the market.*
- $S = \{S_u | u \in U\}$ *is the set of unneeded item lists of users.* $S_u \subseteq I$ *is the set of items that unneeded by user* u.
- $W = \{W_u | u \in U\}$ *is the set of wish lists of users.* $W_u \subseteq I$ *is the set of wanted items of user* u.

The elementary exchange behaviour in the market is the **swap**, denoted as $[(u, i), (v, j)]$, subject to $i \in S_u \cap W_v$ and $j \in S_v \cap W_u$. It means that user u use the item i to trade user v's item j. The swap cover based on a simple exchange market is a set of swaps C. It is **conflict-free** if $\forall u, i \in S_u$, swap $[(u, i), (*, *)]$ appears at most once in C, where the first $*$ is any other user $v \neq u$ and the second $*$ is any item. For example, if $[(u, i), (v, j)]$ and $[(u, i), (w, k)]$ appear together, a conflict is caused since it is not feasible for u to give item i to two users.

The problem is to find a conflict-free swap to maximize the number of items being exchanged. Its decision problem is defined as following:

Definition 2.7. ***SimpleMarket*** *Given a simple exchange market* (U, I, S, W)*, determine if there exists a conflict-free swap cover with number of items exchanged* $\geq K$*.*

Unfortunately, the problem is NP-hard even in the simple exchange market:

Theorem 2.7. *SimpleMarket is NP-Complete.*

The next model we consider is the probabilistic exchange markets. This model improve the simple exchange market by adding a probability setting to describe the social connection and personal income/outcome matching. Formally, this model is defined as below:

Definition 2.8. ***Probabilistic Exchange Markets*** *The simple exchange market is a tuple* $(U, I, S, W, P_u(v), Q_u(i, j))$*.*

- $U = \{u_1, \ldots, u_n\}$ *is the set of users in the market.*
- $I = \{i_1, \ldots, i_m\}$ *is the set of items in the market.*
- $S = \{S_u | u \in U\}$ *is the set of unneeded item lists of users.* $S_u \subseteq I$ *is the set of items that unneeded by user* u*.*
- $W = \{W_u | u \in U\}$ *is the set of wish lists of users.* $W_u \subseteq I$ *is the set of wanted items of user* u*.*
- $P_u(v)$ *denote the probability that* u *is willing to do exchange with* v*.*
- $Q_u(i, j)$*, where* $i \in S_u$ *and* $j \in W_u$*, denotes the probability that* u *is willing to exchange item* i *with item* j*.*

We also consider a more complex exchange behaviour, the cycle exchange. The cycle exchange, denoted as $[(u_1, i_1), (u_2, i_2), \ldots, (u_l, i_l)]$, means that u_1 gives item

i_1 to u_2, and u_2 gives i_2 to u_3, ..., u_l gives i_l to u_1. The probability of a cycle being realized is:

$$P_{u_1}(u_2) \times Q_{u_1}(i_1, i_l) \times P_{u_2}(u_3) \times Q_{u_2}(i_2, i_1) \times \ldots P_{u_l}(u_1) \times Q_{u_l}(i_l, i_{l-1})$$

In practice, we may wish to limit the length of cycles to maximum of k. We define the cycle cover as a conflict-free set C of cycle exchanges, meaning that any pair (u, i) appears at most once in all exchanges in C. Our aim is to find a cycle cover which maximize the expected number of items being exchanged. Therefore we define the ProbMarket problem:

Definition 2.9. ***ProbMarket*** *Given a probabilistic exchange market* $(U, I, S, W, P_u(v), Q_u(i, j))$*, determine if there exists a conflict-free cycle cover whose expected number of items exchanged* $\geq K$*.*

Not surprisingly, this is also an NP-Complete problem:

Theorem 2.8. *ProbMarket is NP-Complete.*

The simple/probabilistic exchange markets can be represented as a graph G. For each user u, we create one node in G labeled u. For each item $i \in S_u \cap W_v$, we create a directed edge from u to v labeled i. A swap is a graphic cycle of length 2. An exchange cycle shorter than k is a graphic cycle of length up to k. A conflict-free cycle(swap) cover, is a set of cycles (swaps) with no common edges. In the simple exchange markets, the weight of a cycle is the number edges in it. In the probabilistic exchange markets, the weight of a cycle is the expected number of elements exchanged in the cycle based on $P_u(v)$ and $Q_u(i, j)$. The problem of finding a conflict-free cycle (swap) cover with length limitation k becomes finding a conflict-free cycle (swap) cover shorter than k with maximum sum of weights in the graph.

Based on the graph representation, four different algorithms are designed to find the conflict-free cycle cover in the graph:

- **Maximal Algorithm** This algorithm repeatedly runs a breath first search from a randomly selected node, find a new cycle and remove the cycle from the graph until no cycle exists in the graph. Then the cycles found in these iterations form a conflict-free cover. The algorithm runs for M rounds and M random conflict-free cycle covers are found. The one with maximum weight is selected as the result.

- **Greedy Algorithm** This algorithm repeatedly finds the maximum weighted cycle in the current graph and remove it until no cycle exists in the graph. The cycles found in these iterations form a conflict-free cover, which is returned as the result.

- **Local Search Algorithm** This algorithm starts with an empty conflict-free cover. It iteratively finds a random cycle that is not ever picked, tries to add it into the current cover and remove any existing cycles with conflict. If the new cover is better than the current cover, then the current cover is replaced with the new cover. The algorithm stops until no improvement can be made and the current cover is returned as the result.

- **Greedy/Local Search Algorithm** This algorithm differs from local search algorithm in only one respect: instead of starting with an empty cover, the greedy/local search algorithm starts with an initial cover which is the output of the greedy algorithm. Then local search improvement is done like the local search algorithm.

Based on analysis in [8], maximal algorithm has no obvious approximation bound; greedy algorithm is a $2k$-approximation; local search algorithm is a $2k-1$-

approximation; greedy/local search algorithm is a $2(2k+1)/3$-approximation. The empirical study shows that the accuracy of maximal algorithm has comparable to that of other algorithms.

2.1.4 Overview of Exchange Models

In this subsection, we summarize the models that we previously introduced and show the relationships among them.

All the models can be generally classified as allocation models and exchange models. In the allocation model, there is no initial connection between the agents (patients / users) and resources (kidneys / items), while in the exchange models the initial endowments play an important role in the problem. In all the models that we introduce, the house allocation is the only allocation model and the other models are the exchange model.

Although the models are designed for various purposes, some of the models are closely related. The house marketing and the paired kidney exchange with strict preference are equivalent. By substituting "patient" for "agent" and "donor" for "house", the house marketing problem becomes the paired kidney exchange problem. Moreover, as explained in [8], the CSEM can also be applied on multi-way kidney exchange problem with 0-1 preference.

A centralized algorithm which outputs the matching between agents and resources is called a mechanism. According to the nature of the market, good mechanism needs to be Pareto-efficient and strategy-proof. For exchange models, it is interesting to find individual rational matching, the core matching or a competitive equilibrium[3]. The top trading cycle, which is a mechanism applied on both house marketing and paired exchange with strict preference, achieves Pareto-efficient and

[3]We are not interested in finding competitive equilibrium for kidney exchange because price the kidney is illegal.

strategy-proof and always outputs a matching in the core. When the list kidney exchange is allowed, a variant of the top trading cycle, called the TTCC, is used and also achieves all the good properties when the proper chain selection rule is used.

Fairness is another concern. For house allocation, any Pareto-efficient mechanism is proven to be dictatorship, which means no mechanism is absolutely fair. For two-way paired kidney exchange with 0-1 preference, lottery mechanism is used to ensure the fairness. Lorenz-dominance defines the fairest lottery mechanism, and this mechanism is found for two-way kidney exchange with 0-1 preference.

Other research focuses on the global utility. For example, multi-way kidney exchange is proposed to maximize the patients been covered. However, several problems on finding multi-way kidney exchange are proven to be NP-complete or even inapproximable. In [8], the algorithms also aim at maximizing the global number of item exchanged, but the other properties such as strategy-proofness, competitive equilibrium and the core are not considered.

2.2 Recommender System

The recommender system is a broad term describing any system that produces individually recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options[19]. It originates from extensive work in different areas, such as cognitive science[39], approximation theory[46], information retrieval[45] and forecasting theories[13]. From mid 1990's, it becomes an independent research area focusing on the recommendation problems which are based on ratings structure. The most common problem in the recommender system is to suggest a list of items (e.g.

restaurant, house and movie) or social element (e.g. friend, event or group) which the user might like most. This problem is often reduced to predict the "rating" or "preference" that a user would give to the item[11]. Formally speaking, there is a function describing the rating that users would give to items:

$$R : USER \times ITEM \mapsto RATING \tag{2.1}$$

Here the $USER$ is the set of users in the system (e.g. the buyer in an online store), and the $ITEM$ is the set of all possible items that can be recommended. The $RATING$ is a totally ordered set denoting the set of possible ratings that a user can give to an item. Possible ratings can be binary (e.g. like/dislike), discrete values (e.g. one to five stars) or continuous real values. Based on R, we could recommend one item i_u for each user u which maximizes the rating function:

$$i_u = argmax_{i \in I} R(u, i_u)$$

Sometimes, instead of choosing only one item, k items are required for each user. This is also known as the top-k recommendation[47].

The central problem of recommender system is that the rating function R is not fully known to the system. The system only knows the ratings that users have already given to the items. This means the recommender system must predict the function R, based on the existing known ratings and other background information, such as user profiles, purchase histories and search logs.

According to [11], the recommender system can be classified into the following categories based on the techniques used:

- **Content-based Recommendation** [36, 30, 34]The user is recommended items solely based on the content of items. The content of an item is the

information describing the item. For example, in a movie recommender system, a movie's content contains its title, genre, description and etc; in a news recommender system, the headline and body are the content belonging to a piece of news. A typical content-based recommender system works as follow:

1. Process content of items and construct a representation for content of each item. For example, a text-based item (e.g. web page, book and news) can be represented as some informative words[36] or represented as a vector[30, 34, 14].
2. Learn a model for each user based on her past feedbacks and the item's content. The model is learned from the past ratings that the users give to the items. Various IR and machine learning techniques are employed to learn the model, including the Rocchio's algorithm[30, 36, 14], Bayesian classifier[36, 34], nearest neighbor algorithm[36], PEBLS[36], Decision trees[36] and neural nets[36].
3. Predict users' rating of unseen items based on the model and recommend to the users.

The content-based recommendation has a few limitations: 1) it only works well when the content of items are easily analyzed by a computer (e.g. text). It is hard to be applied on multimedia data, such as image, audio and video. 2) The recommendation can be over-specialized. The recommendations highly rely on what the user rated in the past. As a result, the recommendations are over-focused on a small cluster. It is not able to bring the user some interesting options that she has never tried. For example, a user never reads science fiction will never be recommended with any science fiction book, no matter how popular the book is. 3) New users with few ratings may not get

high-quality recommendations. The system can accurately model the user's preference only after sufficient ratings are made.

- **Collaborative Recommendation** The user is recommended items purely based on what other people with similar preference chose in the past. In collaborative recommender system, the content of the item is not important. The score is only predicted based on how other users rate the item. There are two classes of collaborative filtering methods: the **memory-based algorithm** and the **model-based algorithm**.[18] The memory-based algorithm predicts the rating directly from the entire database.[38, 18, 25, 35] To predict the rating for a user, the system find some other (usually top-K) users that are similar to the current user. These users are called neighbours. The ratings of neighbours are aggregated to generate the prediction of the current user. Unlike the memory-based algorithm, the model-based algorithm firstly learns a model using the database collection with data mining and machine learning techniques.[18, 35, 15, 29, 33] Then the ratings are predicted by applying the model. Various learning techniques are used for collaborative recommendation. In [15], the problem is modeled as a classification problem and classification algorithms such as the Singular Vector Decomposition are used. In [33], the Latent Dirichlet Allocation is used to model the problem and EM algorithm is used for model fitting. In [29], authors try to embed the users' interest and items' features into a high dimensional linear space and matrix factorization techniques are used to find the embedding. No matter which approach is used, a pure collaborative recommender system only considers the rating relationship between the users and the items. The content of an item is not used while finding the neighbours and building the model. The collaborative recommendation also has its own limitations: 1) new items,

which have very few ratings, may not be recommended to users, no matter how high its rating is and how it fits a user's need. 2) New users with very few ratings may not get correct recommendation. This limitation also exists in content-based recommendation. 3) Critical mass of users is needed for high-quality recommendation. For example, a user with very odd taste may not get accurate recommendation because there is no other user with similar taste as her.

- **Hybrid Approaches** These methods combines both content-based and collaborative recommendation. The hybrid approach helps to overcome the limitations in content-based and collaborative recommendation. There are four ways to combine the two approaches:

  1. Implementing collaborative and content-based methods as two individual model and making prediction by combining their output. For example, [22] use a linear combination. The weight assigned to both methods are adjusted according to the user feedback. [16] uses a switching-based combination. While predicting rate for an unseen item, the system switch to content-based or collaborative recommender according to the pre-defined rule.

  2. Adding content-based features to collaborative modules. For example, in [37], a "collaborative via content" approach is used. It creates content-based profile for each user and uses it to calculate correlation between users. Therefore, two users are considered as similar users not only if they have rated the same item, but also if they have rated similar items based on content.

  3. Adding collaborative features to content-base models. For example, in

[24] the authors consider using the social connection between users to adjust the feature weighting in vector-based representation of item content.

4. Building a single model considering the content and collaboration simultaneously. For example, in [12] a statistic model considering the user profiles and the item characteristics is proposed. The model is trained using Markov chain Monte Carlo method with the past rating data.

As a research area, the recommender system has been extensively studied and various techniques are proposed. However, the item-exchange recommender system proposed in this thesis is not a typical recommender system. Like the traditional recommender system, our system also aims at recommending users with items that maximize their utility function. But the main goal of our system is not predicting the hidden utility function, but computing it efficiently. Therefore, the technique used in this thesis is not related to a traditional recommender system. For this reason, we do not survey all the recommender system techniques.

2.3 Summary

In this chapter, we review the existing research work related to this thesis. In the first part of this chapter, we survey the exchange economic models. These exchange models are mathematical tools for analysis and simulation of a certain type of exchange activities. We review related work on the house allocation and exchange models, kidney exchange models and the CSEM. We summarize the proposed models, algorithm/mechanisms and their characteristics from both economics and computer science community. In the second part of this chapter, we review some research work on the recommender system. The recommender system pro-

vides users with personalized suggest on items. The major challenge of recommender system is to predict a hidden preference function based on the past ratings that users have given to the items. The techniques are classified as three types: the content-based, collaborative and the hybrid approach. The content-based methods only make recommendation based on the content of item (e.g. description, title), while the collaborative methods only recommend based on other users' choices. The hybrid method combines the two methods to overcome some limitation in pure content-based and collaborative method and leads to a better performance.

CHAPTER 3

PROBLEM FORMULATION AND PRELIMINARIES

In this chapter, we propose the Binary Value-based Exchange Model (BVEM), which models the basic user exchange bahaviour in the community system. Based on BVEM, we define the Top-K Exchange Pair Monitoring Problem, which is the major problem we should solve in our recommendation system. Then we prove the problem is NP-hard.

3.1 Problem Definition

In the community system, we assume that there are n users $U = \{u_1, u_2, \ldots, u_n\}$, and m items $O = \{I_1, I_2, \ldots, I_m\}$. Each user u_i has two item lists, the unneeded item list L_i and the wish list W_i. Each item I_j is labelled with a tag v_j as its public price. Given a group of items $O' \subseteq O$, the value of the item set is the sum on the prices of all items in O', i.e. $V(O') = \sum_{I_j \in O'} v_j$. In the example for Figure 1.1 and Figure 1.2, the value of the item set $V(\{I_1, I_2, I_3\})$ =\$100 according to the price list in the figures.

In this thesis, we adopt the Binary Value-based Exchange Model as the under-

lying exchange model in the community system. Given two users u_i and u_l, as well as two item sets $S_i \subseteq L_i$ and $S_l \subseteq L_l$, an exchange transaction $E = (u_i, u_l, S_i, S_l)$ represents the deal that u_i gives all items in S_i to u_l and receives S_l in return. The gain of the exchange E for user u_i is measured by the total value of the items he receives after the exchange, i.e. $G(E, u_i) = V(S_l)$. Similarly, the gain of user u_l is $G(E, u_l) = V(S_i)$. This exchange is eligible under BVEM with relaxation parameter β $(0 < \beta \leq 1)$, which follows the formal definition below.

Definition 3.1. ***Eligible Exchange Pair***

The exchange transaction $E = (u_i, u_l, S_i, S_l)$ is eligible, if it satisfies 1) Item matching condition: $S_i \subseteq W_l$ and $S_l \subseteq W_i$; and 2) Value matching condition: $\beta V(S_i) \leq V(S_l) \leq \beta^{-1} V(S_i)$.

Assuming that all users in the system are rational, each user u_i always wants to maximize his gain in the exchanges with other users. In the following, we prove the existence of a unique optimal exchange among all exchanges between u_i and u_l, maximizing both of their gains.

Lemma 3.1. *For any pair of users, u_i and u_l, there exists a dominating exchange pair $E = (u_i, u_l, S_i, S_l)$ such that for any $E' = (u_i, u_l, S_i', S_l')$ the following two events can never happen: 1) $G(E', u_i) > G(E, u_i)$, or 2) $E(E', u_l) > G(E, u_l)$.*

Proof. We prove this lemma by construction and contradiction. We order all eligible exchange pairs with non-increasing order on $G(E, u_i)$. For all exchange pairs with exactly the maximal gain for u_i, we further find the unique exchange pair $E = (u_i, u_l, S_i, S_l)$ by maximizing the gain for u_l. If E does not satisfy the condition in the lemma, there are two possible cases. In the first case, there exists an exchange pair E' that $G(E', u_i) > G(E, u_i)$. Depending on our construction method, this situation can never occur. In the second case, u_l has a better option with higher gain

in $E' = (u_i, u_l, S'_i, S'_l)$, i.e. $G(E', u_l) = V(S'_i) > G(E, u_l) = V(S_i)$. If this happens, we will show in the following that $E''(u_i, u_l, S'_i, S_l)$ is also an eligible exchange pair, thus violating the construction principle of E. Based on the definition of eligible exchange pair, we know that

$$G(u_i, E') = V(S'_l) \geq \beta V(S'_i) = \beta G(u_l, E')$$

Since $G(u_i, E)$ is the maximal gain of u_i on any exchange pair, it is easy to verify that $V(S_l) \geq V(S'_l) \geq \beta V(S'_i)$. On the other hand, it can be derived that

$$V(S_l) \leq \beta^{-1} V(S_i) \leq \beta^{-1} V(S'_i)$$

Combining the inequalities, we conclude $E'' = (u_i, u_l, S'_i, S_l)$ is also eligible. Moreover, $G(u_i, E'') = V(S_l) = G(u_i, E)$ and $G(u_l, E'') = V(S'_i) > V(S_i) = G(u_l, E)$, which also violate our construction method. This contradiction leads to the correctness of the lemma. □

The lemma suggests the existence of an optimal exchange solution between u_i and u_l for both parties, denoted by $E^*(u_i, u_l)$. However, for each user u_i, there may exist different eligible exchange pairs with different users at the same time. To suggest more promising exchange pairs to the users, we define *Top-K Exchange Pair* as below.

Definition 3.2. ***Top-K Exchange Recommendations***

For user u_i, the top-k exchange pairs, i.e. $Top(k, i)$, includes the k most valued exchange pairs $E^*(u_i, u_l)$ with k different users.

In the definition above, each pair of user (u_i, u_l) contributes at most one exchange pair to $Top(k, i)$. It is because there is a dominating exchange plan be-

ID	Name	Price
I_1	Nail	\$10
I_2	Ribbon	\$20
I_3	Screwer	\$70
I_4	Hammer	\$80
I_5	Paint	\$100
I_6	Drill	\$170

Time	Operation	User	Wish list	Unneeded list	Top-1	Top-2
1		u_1	I_2	I_1,I_4	--	--
		u_2	I_1,I_6	I_4,I_5	(u_2, u_3, {I_6}, {I_4, I_5})	--
		u_3	I_4,I_5	I_2,I_3,I_6	(u_3, u_2, {I_4, I_5}, {I_6})	--
2	Insert I_3 into W_1	u_1	I_2,**I_3**	I_1,I_4	**(u_1, u_3, {I_2, I_3}, {I_4})**	--
		u_2	I_1,I_6	I_4,I_5	(u_2, u_3, {I_6}, {I_4, I_5})	--
		u_3	I_4,I_5	I_2,I_3,I_6	(u_3, u_2, {I_4, I_5}, {I_6})	**(u_3, u_1, {I_4}, {I_2, I_3})**
3	Delete I_5 from U_2	u_1	I_2,I_3	I_1,I_4	(u_1, u_3, {I_2, I_3}, {I_4})	--
		u_2	I_1,I_6	I_4	--	--
		u_3	I_4,I_5	I_2,I_3,I_6	(u_3, u_1, {I_4}, {I_2, I_3})	--

Figure 3.1: Running Example of Top-K Exchange Pair Monitoring with $\beta = 0.8$

tween two users u_i and u_l. Therefore, it is less meaningful to output two different exchange suggestions between a single pair of users. The main problem we want to solve in this thesis is to provide an efficient mechanism to monitor top-k exchange recommendations for each user in real time.

Problem 3.1. ***Top-K Exchange Pair Monitoring***

For each insertion or deletion on any item list L_i and W_i for user u_i, update the $Top(k, j)$ for every user u_j in the system.

Upon insertions or deletions on the item lists of user u_i, the top-k exchange pairs of u_i or other users is subject to change. Figure 3.1 shows an example to help understand the impact of item updates. At the initial timestamp, there is only one eligible exchange pair between u_2 and u_3, i.e. $(u_2, u_3, \{I_6\}, \{I_4, I_5\})$. The gain of u_3 in this potential exchange is 180\$. At the second timestamp, assume that no exchange is happened and a new item I_3 is inserted into u_1's wish list. The exchange pair between u_1 and u_3 becomes eligible, as is listed in the table. The gain of u_3 from the new exchanging pair is \$80, which is smaller than her gain from the previous exchange suggestion with u_2. As a result, the new exchanging pair is the second best recommendation for u_3. At time 3, I_5 is deleted from unneeded list of u_2. This breaks the existing eligible exchang pair between u_2 and u_3, and there is no other eligible exchange pairs between them. Therefore, this exchange pair is deleted from the recommendation list of both users. It is important to note that our system only presents the suggestions to the users, but never automatically

commits these exchanges.

In the following theorem, we prove that the computation of top-1 exchange pair is NP-hard, even when there are only two users in the system.

Theorem 3.1. *Given two users u_i and u_l, finding the optimal eligible exchange pair between u_i and u_l is NP-hard.*

Proof. We reduce the *Load Balancing Problem* to our problem. Given a group of integers $X = \{x_1, x_2, \ldots, x_n\}$, the problem of load balancing is deciding if there exists a partition $X_1 \subset X$ and $X_2 \subseteq X$ ($X_1 \cap X_2 = \emptyset$ and $X_1 \cup X_2 = X$) that $\sum_{x_i \in X_1} x_i = \sum_{x_j \in X_2} x_j$. Load balancing problem is one of the most famous NP-Complete problems [52].

Given each instance of load balancing problem, i.e. X, we construct the item lists for u_i and u_l as follows. For each $x_j \in X$, a corresponding item I_j is constructed with value $v_j = x_j$. All these items I_j ($1 \leq j \leq n$) are inserted into the wish item list W_i for u_i and unneeded item list L_j. A new item I_{n+1} is then created with value $v_{n+1} = \sum_{x_j \in X} x_j/2$. We insert I_{n+1} into L_i and W_j. This reduction can be finished in $O(n)$ time. By setting $\beta = 0$, our problem tries to find a subset in W_i with the exact total value as I_{n+1}. If such a solution is always discovered by some algorithm in polynomial time, load balancing problem is also solvable in polynomial time. If this is the case, we will prove P=NP because load balancing problem is NP-Complete. Therefore, our problem is NP-Hard. □

The last theorem shows that the complexity of finding top-k exchange pair between any two users is exponential to the size of the item lists. Fortunately, the number of items owned by the users is usually limited in most of the online community systems. This partially relieves the problem of optimal exchange pairing. Therefore, the major problem to overcome for top-k exchange pair monitoring is

Notation	Description
$U = \{u_i\}$	the set of users in the community
$O = \{I_j\}$	the set of items with all users
L_i	the unneeded item list for user u_i
W_i	the wish list for user u_i
v_j	the value of the item I_j
$V(O')$	the value of an item set $O' \subseteq O$
S_i S_l	item subset of L_i and L_l respectively
$E(u_i, u_l, S_i, S_l)$	exchange pair between u_i and u_l
$G(E, u_i)$	the gain of u_i from exchange E
β	relaxation factor on value matching condition
$E^*(u_i, u_l)$	the optimal exchange pair between u_i and u_l
AVT	approximate value table
$AVT[m]$	mth entry in AVT
N	maximal number of items in any list
ϵ	approximation bound
$v_{\min}, v_{\max}$	minimal and maximal value of any item combination
$\mathcal{N}$	maximal number of entries in any AVT
$Top(k, i)$	Top-k exchanges list for user u_i
θ_i	minimal value of exchange pairs in $Top(k, i)$
$UL(I_j)$	set of users who have I_j in their unneeded item list
$CL(I_j)$	set of users who have I_j in their critical item set
κ	number of top results to be calculated initially
κ_i	number of top results u_i currently keep
K_i	critical item sets for user u_i

Table 3.1: Table of Notations

how to effectively select some pairs of users to re-calculate the optimal exchange, when some insertion or deletion occurs. In the rest of the thesis, we present our data structure for indexing the possible exchange pairs, in the presence of frequent list updates.

3.2 Notations

For ease of reading, all of the notations are summarized in Table 3.1.

3.3 Summary

In this chapter, BVEM, a model taking care of both item combination and price, is adopted as the underlying exchange model. Based on this model, the Top-K Exchange Recommendation is defined, which recommend users the best valued exchange pairs with respect to exchange eligibility. To compute and maintain the Top-K Exchange Recommendation in a real-time scenario, the Top-K Exchange Pair Problem is defined. This problem is proven to be NP-Hard.

CHAPTER 4

COMPUTING EXCHANGE PAIRS

In this chapter, we present our solution to the Top-k Exchange Pair Monitoring problem. We solve this problem in two steps: 1) we solve the T1U2 exchange problem, which is a special case in which only exchange pair between two users are computed; 2) we solve the general problem, based on our T1U2 exchange algorithm.

4.1 Exchange between Two Users

In this section, we focus on a special case of the exchange recommendation problem, with only two users in the system are looking for the best valued exchange pair between them. Later we will extend our discussion to the general case with arbitrary number of users. For simplicity, we call it a *T1U2 Exchange*. Algorithmically, T1U2 exchange can be solved by an offline algorithm with exponential complexity in term of the list sizes.

The offline algorithm works as follows. It first computes the intersections between the wish list and unneeded item list, i.e $W_i \cap L_l$ and $L_i \cap W_l$. Then all the subsets of the two temporary lists are enumerated. The algorithm tests every pair of the subsets to find the pairing satisfying Definition 3.1 and maximizing the gain

Algorithm 4 Brute-force algorithm for T1U2 exchange(L_i, W_i, L_l, W_l)

```
Clear optimal solutions S*
Generate subsets φ_L = 2^{L_i ∩ W_l} and sort on value
Generate subsets φ_R = 2^{L_l ∩ W_i} and sort on value
Set m = |φ_R|
for n from |φ_L| to 1 do
    while m > 0 and β * |φ_R[m]| > |φ_L[n]| do
        m = m − 1
    end while
    if φ_L[n] and φ_R[m] is an eligible exchange then
        S* = (u_i, u_l, φ_L[n], φ_R[m]) if V(φ_L[n]) ≥ G(S*, u_i) and V(φ_R[m]) ≥ G(S*, u_l)
    end if
end for
Return S*
```

of both users. Details about this algorithm is illustrated in Algorithm 4. The running time of this algorithm is exponential to the list size, i.e. $O(|S_i|2^{|S_i|} + |S_l|2^{|S_l|})$. Unfortunately, there does not exist any exact algorithm with polynomial complexity, unless P=NP. Hence it is more interesting to find some alternative solution, outputting approximate results with much better efficiency.

Definition 4.1. ***ϵ-Approximate T1U2 Exchange for*** u_i

Assuming $E^* = (u_i, u_l, S_i, S_l)$ is the highest valued exchange pair between user u_i and u_l, an exchange pair, $E' = (u_i, u_l, S_i', S_l')$, is said to be ϵ-approximate for u_i if the gain is no worse than E^* by factor $1 - \epsilon$, i.e. $G(E', u_i) \geq (1 - \epsilon)G(E^*, u_i)$.

Unlike exact top-1 exchange pairing, ϵ-approximate exchange does not exhibit similar property as in Lemma 3.1. An ϵ-approximate exchange pair for u_i may not be ϵ-approximate for u_l. Therefore, the computation involving u_i and u_j may return different results to the users.

Inspired by the famous polynomial-time approximation algorithm on the subset sum problem [23], we design a fully polynomial-time approximation scheme(FPTAS) to calculate ϵ-approximate T1U2 exchange. Moreover, we show how to utilize the

solution to design a reusable index structure to support updates.

The approximation scheme follows similar idea in the FPTAS on subset sum problem. Generally speaking, the original brute-force algorithm spends most of the time on generating all the item combinations of $W_i \cap L_l$ and $L_i \cap W_l$. There are many redundant combinations, which share almost the same value with others. In the new algorithm, it only generates some of the combinations of the items in $W_i \cap L_j$ and $L_i \cap W_j$. These combinations are maintained in a table indexed by their approximate values. Other item combinations are merged into the table when their value is similar to the existing ones. In particular, given the approximation factor ϵ, the exact value of an item set, $V(O')$, is transformed to some approximate value, $\gamma(O')$, guaranteeing that

$$V(O') \leq \gamma(O') \leq (1-\epsilon)^{-1} V(O') \tag{4.1}$$

We hereby utilize the following rounding function $f(x)$. Here, $v_{\max}$ and $v_{\min}$ are the maximal/minimal values of any item combination, ϵ is error tolerance and N is the maximal number of items.

$$f(O') = \left\lceil \frac{\log v_{\min} - \log V(O')}{\log\left(1 - \frac{\epsilon}{N}\right)} \right\rceil \tag{4.2}$$

Intuitively, $f(O')$ is the minimal integer m that $v_{\min}\left(1-\frac{\epsilon}{N}\right)^{-m}$ $\geq V(O')$. Since $v_{\min} \leq V(O') \leq v_{\max}$ and $f(O')$ always outputs an integer, $f(O')$ can only be a non-negative integer between 0 and $\mathcal{N} = \lceil (\log v_{\min} - \log v_{\max}) / \log(1 - \frac{\epsilon}{N}) \rceil$. Based on this property, we implicitly merge the item combinations to $\mathcal{N}$ groups, i.e. $\{S_1, S_2, \ldots, S_{\mathcal{N}}\}$. Each group S_m contains every item combination O' with $f(O') = m$, i.e. $S_m = \{O' | f(O') = m\}$. For every item combination $O' \in S_m$, we have the common approximate value $\gamma(O')$ for O', i.e. $\gamma(O') = v_{\min}\left(1-\frac{\epsilon}{N}\right)^{-m}$, which satisfies Equation (4.1).

Algorithm 5 AVT **Generation** (Item set O', Error bound ϵ , maximal value $v_{\max}$, minimal value $v_{\min}$, maximal item number N)

Generate an empty approximate value table AVT
Create a new entry $AVT[0]$
Set $AVT[0].lbi = \emptyset$
Set $AVT[0].ubi = \emptyset$
Set $AVT[0].value = 0$
Set $AVT[0].lb = AVT[0].ub = 0$
for each item $I_j \in O'$ **do**
  for each entry $AVT[m] \in AVT$ **do**
    Calculate $M = f(AVT[m].value + v_j)$
    if there is $AVT[n].value = M$ **then**
      if $AVT[m].lb + v_j < AVT[n].lb$ **then**
        Update $AVT[n].lb$ and $AVT[n].lbi$
      end if
      if $AVT[m].ub + v_j > AVT[n].ub$ **then**
        Update $AVT[n].ub$ and $AVT[n].ubi$
      end if
    else
      Create a new entry $AVT[n]$ in AVT
      $AVT[n].value = M$
      $AVT[n].lb = AVT[m].lb + v_j$
      $AVT[n].ub = AVT[m].ub + v_j$
      $AVT[n].lbi = AVT[m].lbi \cup \{I_j\}$
      $AVT[n].ubi = AVT[m].ubi \cup \{I_j\}$
    end if
  end for
end for
Return AVT

These groups are maintained in a relational table, called *Approximate Value Table* (or AVT in short). In AVT, each entry $AVT[m]$ records some statistical information of the group S_m, to facilitate the computation of ϵ-approximate T1U2 exchange. Specifically, we use $AVT[m].value$ to denote the common approximate value of all item combinations in S_m. We use $AVT[m].lb$ ($AVT[m].ub$ resp.) to denote the lower bound (upper bound resp.) of all the item combinations in S_m. We also keep the item combinations achieving the lower bound and upper bound, i.e. $AVT[m].lbi$ and $AVT[m].ubi$. In Table 4.1, we present an example of AVT.

Entry	approximate value	lb	lbi	ub	ubi	All item combinations
$AVT[1]$	2	2	$\{I_1\}$	2	$\{I_1\}$	$\{I_1\}$,$\{I_2\}$
$AVT[2]$	4	3	$\{I_3\}$	4	$\{I_1, I_2\}$	$\{I_3\}$,$\{I_1, I_2\}$
$AVT[3]$	8	5	$\{I_1, I_3\}$	7	$\{I_1, I_2, I_3\}$	$\{I_1, I_3\}$,$\{I_2, I_3\}$,$\{I_1, I_2, I_3\}$

Table 4.1: Example of approximate value table on a 3-item set

To construct the AVT table, we sort all items based on their identifiers. At the beginning, the algorithm initializes the first entry $AVT[0]$ in the table. We set $AVT[0].value = AVT[0].lb = AVT[0].ub = 0$, empty $AVT[0].lbi$ and $AVT[0].ubi$ at the same time. For each item I_j in the input item set O', the algorithm iterates every through existing entry $AVT[m]$ in the AVT and updates as follows. For every entry $AVT[m]$, our algorithm tries to generate a new entry $AVT[n]$ with $n = f(AVT[m].value + v_j)$. If $AVT[n]$ already exists, it tries to merge I_j into $AVT[m].lbi$ and $AVT[m].ubi$, checking if they can generate new lower and upper bound for group S_n. If $AVT[n]$ does not exist in the table, a new entry is created. The details are available in Algorithm 5.

If we run the algorithm on a 3-item set $O' = \{I_1, I_2, I_3\}$ with item prices $v_1 = 2$, $v_2 = 2$ and $v_3 = 3$, the result AVT is presented in Table 4.1, with $(1 - \epsilon/N)^{-1} = 2$ and $v_{\min} = 1$. There are 7 non-empty combinations in O', including $\{I_1\}$, $\{I_2\}$, $\{I_3\}$, $\{I_1, I_2\}$, $\{I_1, I_3\}$, $\{I_2, I_3\}$ and $\{I_1, I_2, I_3\}$. After finishing the construction of the AVT table, there are only 3 entries in the table, which is much smaller than than the original number of item combinations. The information of the groups are all listed in the rows of the table. We also include the concrete item combinations in the last column for better elaboration, although AVT does not maintain them in the computation.

In the following lemma, we show that the output AVT summarizes every item combination within error bound ϵ.

Lemma 4.1. *Given any item set O', for each item combination $O'' \subseteq O'$, the AVT table calculated by Algorithm 5 contains at least one entry $AVT[m]$ that*

$$V(O'') \geq (1-\epsilon)AVT[m].value$$

$$AVT[m].lb \leq V(O'') \leq AVT[m].ub$$

Proof. For simplicity, let $\delta = 1 - \epsilon/N$. We apply mathematical induction to that, $\forall O'' \in O'$, there is an $AVT[n]$ such that:

$$V(O'') \geq \delta^{|O''|} AVT[m].value \tag{4.3}$$

$$AVT[m].lb \leq V(O'') \leq AVT[m].ub \tag{4.4}$$

Basically, if $|O''| = 0$, namely $O'' = \emptyset$, the Equation 4.3 and 4.4 hold by giving $AVT[0]$.

Then we inductively prove the lemma. Assume that the the Equation 4.3 and 4.4 hold for all $|O'''| = k$, we are going to prove that they also hold for O'' with length $k+1$. Let $O'' = \{I_1, I_2, \ldots, I_{k+1}\}$. By the assumption, for $O''' = \{I_1, I_2, \ldots, I_k\}$, there is a $AVT[n]$ such that Equation 4.3 and 4.4 holds. According to line 9-12 in Algorithm 5, the AVT table is updated according to I_{k+1} and $AVT[n]$. Let the updated (line 11-14) or new created (line 16-21) AVT entry be $AVT[m]$. We can verify that:

$$\begin{aligned} V(O'') &= V(O'' - I_{k+1}) + v_{k+1} \\ &\geq \delta^k AVT[n].value + v_{k+1} \\ &\geq \delta^k (AVT[n].value + v_{k+1}) \\ &\geq \delta^{k+1} f(AVT[n].value + v_{k+1}) \\ &= \delta^{k+1} AVT[m].value \end{aligned}$$

$$\begin{aligned} V(O'') &= V(O'' - I_{k+1}) + v_{k+1} \\ &\geq AVT[n].lb + v_{k+1} \\ &\geq AVT[m].lb \end{aligned}$$

$$\begin{aligned} V(O'') &= V(O'' - I_{k+1}) + v_{k+1} \\ &\leq AVT[n].ub + v_{k+1} \\ &\leq AVT[m].ub \end{aligned}$$

Since $\delta^k \geq \delta^N = (1 - \epsilon/N)^N \geq 1 - \epsilon$, Lemma 4.1 holds. □

The size of AVT is no larger than $\mathcal{N}$. Therefore, the complexity of the AVT construction algorithm is $O(\mathcal{N}^2|O'|)$. Assuming $v_{\max}$, $v_{\min}$, ϵ and N are all known constants, the algorithm finishes in linear time with respect to the item size $|O'|$, which is supposed to be much faster than the exact algorithm if $\mathcal{N}$ is much smaller

than $2^{|N|}$.

To utilize AVT in T1U2 exchange problem, we create two tables AVT_1 and AVT_2, based on $L_i \cap W_l$ and $W_i \cap L_l$ respectively. If there is an eligible exchange pair between u_i and u_l, the following lemma shows that there must also exist a pair of $AVT[m] \in AVT_1$ and $AVT[n] \in AVT_2$ with close values.

Lemma 4.2. *If $E = (u_i, u_l, S_i, S_l)$ is any eligible exchange and $\epsilon \leq 1 - \beta$, there exists two entries $AVT_1[m] \in AVT_1$ and $AVT_2[n] \in AVT_2$ that*

$$\beta AVT_1[m].lb \leq AVT_2[n].ub \leq \beta^{-1} AVT_1[m].lb$$

$$\beta AVT_2[n].lb \leq AVT_1[m].ub \leq \beta^{-1} AVT_2[n].lb$$

Proof. According to Lemma 4.1, we can find $AVT_1[m]$ and $AVT_2[n]$ such that $AVT_1[m].lb \leq V(S_i) \leq AVT_1[m].ub$, and $AVT_2[n].lb \leq V(S_l) \leq AVT_2[n].ub$. There could be two cases:

- $AVT_1[m].value \geq AVT_2[n].value$
- $AVT_1[m].value < AVT_2[n].value$

These two cases correspond to the two inequalities respectively. We will only prove the first case because of the symmetry.

The left side of the inequations:

$$\begin{aligned} \beta AVT_1[m].lb &\leq \beta V(S_i) \\ &\leq V(S_l) \\ &\leq AVT_2[n].ub \end{aligned}$$

The right side of the inequations:

$$\begin{aligned} AVT_2[n].ub &\leq AVT_2[n].value \\ &\leq AVT_1[m].value \\ &\leq (1-\epsilon)^{-1} AVT_1[m].lb \\ &\leq \beta^{-1} AVT_1[m].lb \end{aligned}$$

So far the first case has been proven. The second case can be proven similarly. □

The last lemma shows that we can find candidate pairs from the approximate value tables, by testing the lower bounds and upper bounds of the entries. Based on the lemma, we present algorithm 6 to show how to discover ϵ-approximate exchange pair for u_i and u_l at the same time. Note that the results for u_i and u_l may not be the same exchange pair. Given the AVT_1 on $W_i \cap L_l$ and AVT_2 on $L_i \cap W_l$, every pair of entries $AVT[m] \in AVT_1$ and $AVT[n] \in AVT_2$ are tested. If the condition in Lemma 4.2 is satisfied, two pairs of eligible exchange pair are generated, i.e. an exchange candidate $(u_i, u_l, AVT[m].ubi, AVT[n].lbi)$ for u_i and another exchange candidate $(u_i, u_l, AVT[m].lbi, AVT[n].ubi)$ for u_l respectively. The algorithm then tests the optimality of the two exchange pairs for u_i and u_l separately. After finding all the eligible exchange pairs, the optimal solutions are returned to u_i and u_l separately.

Theorem 4.1. *Algorithm 6 outputs ϵ-approximate optimal top-k exchange pair between any two users u_i and u_l in linear time.*

Proof. Consider the top-1 eligible exchange (u_i, u_l, S_i, S_l). By Lemma 4.2, we can find an upper (lower) bound item set S_i' in AVT_1, and an lower (upper, *resp.*) bound item set S_l' in AVT_2, such that they form an eligible exchange, and $V(S_i') \geq$

Algorithm 6 Exchange Search on AVT (lists W_i, L_i, W_l, L_l)

1: Clear result set RS_i for u_i and RS_l for u_l
2: Generate AVT_1 on $W_i \cap L_l$ and AVT_2 on $L_i \cap W_l$
3: **for** each pair of entries $AVT_1[m] \in AVT_1$ and $AVT_2[n] \in AVT_2$ **do**
4: **if** $\beta \leq \frac{AVT_1[m].ub}{AVT_2[n].lb} \leq \frac{1}{\beta}$ and $\beta \leq \frac{AVT_2[n].ub}{AVT_1[m].lb} \leq \frac{1}{\beta}$ **then**
5: Generate $(u_i, u_l, AVT[m].ubi, AVT[n].lbi)$ for u_i and $(u_i, u_l, AVT[m].lbi, AVT[n].ubi)$ for u_l
6: Update RS_i and RS_l if necessary
7: **end if**
8: **end for**
9: Return RS_i to u_i and RS_l to u_l

$(1-\epsilon)V(S_i)$, $V(S_l') \geq (1-\epsilon)V(S_l)$. Therefore, (u_i, u_l, S_i', S_l') is an ϵ-approximate top-1 exchange pair. Since both S_i' and S_l' are lower or upper bound item sets, and Algorithm 6 compares all pairs of lower / upper bound values, S_i' and S_j' are guaranteed to be found by Algorithm 6. □

The algorithm to find approximate T1U2 is described in Algorithm 6. Since there are at most $\mathcal{N}$ entries in either table, the time complexity of Algorithm 6 is $O(\mathcal{N}^2)$. By sorting all the entries in decreasing order on approximate value and scanning entries in top-down fashion, we can easily reduce the complexity of the algorithm to $O(\mathcal{N})$.

4.2 General Top-K Exchange

In last section, we use the technique of approximate value table to search top-1 exchange pair between two users u_i and u_l. In real systems, however, there are usually thousands of users online at the same time. To support large community systems for exchange recommendation, we extend our discussion from two users to arbitrary number of users in this section. A straightforward solution to the problem is maintaining $|U|(|U|-1)$ approximate value tables. For each pair of users u_i and

u_l, two approximate value tables AVT_{il} and AVT_{li} are constructed and maintained for item combinations in $W_i \cap L_l$ and $L_i \cap W_l$ respectively. Upon any update of the lists with user u_i, the system re-computes T1U2 between u_i and any other user u_l. $Top(k, i)$ and $Top(k, l)$ are thus updated accordingly with respect to the new optimal exchange between u_i and u_l. Unfortunately, this solution is not scalable in large online community systems on table indexing and maintenance, due to the quadratic number of tables used in this solution.

To reduce the memory space used by the index structure, we do not dynamically maintain approximate value tables between every pair of users. Instead, some lightweight index structure is kept in the system, with space consumption linear to the number of items. Given an update on some list L_i (or W_i) on user u_i, this data structure is used to find out every user u_l with potentially affected $Top(k, i)$ or $Top(k, l)$. To accomplish this, we first derive some necessary condition on top-k exchange pairs, with the concept of *Critical Item Set*.

Definition 4.2. *Given an item list W_i of user u_i, a subset of items $O' \subseteq W_i$ form a critical item set, if $V(W_i) - V(O') < G(u_i, Top(k, i))$.*

In other words, an item set O' is critical to the wish list W_i, if the rest of the items in W_i is of total value no larger than the current optimal gain of u_i. In the following, we use K_i to denote the critical item set on W_i of u_i. Note that Definition 4.2 only provides an sufficient condition on critical item set. Given an item list W_i, there can be hundreds of different combinations of items satisfying the definition above. In Section 4.2.1, we will discuss more on how to construct a good critical item set according to some criterion.

Lemma 4.3. *If $Top(k, i)$ contains an exchange pair $E = (u_i, u_l, S_i, S_l)$, S_i contains at least one item I_j in the critical item set K_i with respect to W_i.*

Proof. Suppose that S_i does not contains any item in K_i. That is, $S_i \subset W_i - K_i$. Therefore, $V(S_i) \leq V(W_i) - V(K_i) < G(u_i, Top(k,i))$. This contradicts the condition that S_i is an top-k exchange. Therefore, S_i contains at least one item in any critical item set. □

Lemma 4.3 implies that the system needs to re-compute the T1U2 exchange between u_i and u_l to update $Top(k,i)$, only if u_l owns at least one critical item of u_i and vice versa. This motivates our index structure based on inverted lists on the critical items. There are two inverted lists on each item, i.e. $CL(I_j)$ and $UL(I_j)$. $CL(I_j)$ consists of a list of users with I_j in his critical item set, and $UL(I_j)$ includes all users with I_j in his unneeded item list.

Generally speaking, when there is an update (insertion or deletion) on W_i of user u_i, the system retrieves a group of candidate users from the inverted lists and computes T1U2 exchange. The candidate set is $\left(\bigcup_{I_j \in W_i} UL(I_j)\right) \bigcap \left(\bigcup_{I_k \in L_i} CL(I_k)\right)$. The detailed description is given in Algorithm 7. By Lemma 4.3, this algorithm does not miss any necessary update on the top recommendation lists. The major cost of the candidate selection is spent on merging the inverted lists on the users. To improve the efficiency of the list merging, every inverted list is sorted on the ids of the users. In the rest of the section, we discuss details on the implementations of some more efficient pruning strategies.

4.2.1 Critical Item Selection

In this part of the section, we resolve the problem on the construction of optimal critical item selection according to Algorithm 7. Given the wish list W_i, there are a large number of different ways to construct the critical item set K_i. Generally speaking, a good critical item set is supposed to reduce the number of candidate users tested in Algorithm 7. To accomplish this, we first derive some cost model

Algorithm 7 General Top-K Update(W_i,u_i)

```
1: Clear the left candidate user set CU_l
2: for each I_j in the critical item set of W_i do
3:    merge UL(I_j) into CU_l
4: end for
5: Clear the right candidate user set CU_r
6: for each I_j ∈ L_i do
7:    merge CL(I_j) into CU_r
8: end for
9: for each u_l ∈ CU_l ∩ CU_r do
10:   Compute T1U2 between u_i and u_l
11:   Update Top(k, i) and Top(k, l) accordingly
12: end for
```

below.

Since $UL(I_j)$ keeps the set of users owning the item I_j in their unneeded item list. Basically, we assume that $|UL(I_j)|$ is relatively small, compared to the total number of users $|U|$, i.e. $|UL(I_j)| \ll |U|$. Moreover, we further assume that $UL(I_j)$ for different items are not strongly correlated. Namely, for any two distinct items I_j and I_k, $|UL(I_j) \cap UL(I_k)| \ll |UL(I_j)|$. With this assumption, the number of candidate users to check, given the critical item set K_i, can be estimated by $\sum_{I_j \in K_i} |UL(I_j)|$.

Based on the analysis above, a good critical item set is equal to the following combinatorial problem with linear constraint.

$$\text{Minimize} : \sum_{I_j \in K_i} |UL(I_j)|$$

$$\text{s.t.} \sum_{I_j \in K_i} v_j \geq V(W_i) - G(u_i, Top(k,i))$$

That is, for an user U_i, we select a set $K_i \subset W_i$, to minimize $\sum_{I_j \in K} |UL(I_j)|$, subject to the sufficient condition $\sum_{I_j \in K} v_j \geq V(W_i) - G(u_i, Top(k,i))$ in Definition 4.2.

User	W_i	L_i	$G(u_i, Top(k,i))$	Critical Item Set
u_1	I_1, I_2, I_3	I_4, I_5, I_6	60	I_1, I_2
u_2	I_2, I_6	I_3, I_5	50	I_6
u_3	I_3, I_5	I_1, I_2, I_6	80	I_5
u_4	I_1, I_4	I_6	0	I_1, I_4
u_5	I_4, I_6	I_1, I_3	10	I_4, I_6

Table 4.2: Example of critical item sets of 5 users

Although this problem is an NP-Complete problem, a near-optimal solution can be obtained by a simple greedy algorithm. Following such construction method, the items in W_i are sorted in decreasing order of $v_j/|UL(I_j)|$. Then the items are selected one by one in this order, until the sum of the value exceeds $V(W_i) - G(u_i, Top(k,i))$.

Table 4.2 shows an example of system with 5 users. The value of the items are $v_1 = 70, v_2 = 40, v_3 = 20, v_4 = 35, v_5 = 80, v_6 = 10$, and $|UL(I_1)| = 3, |UL(I_2)| = 1, |UL(I_3)| = 2, |UL(I_4)| = 1, |UL(I_5)| = 2, |UL(I_6)| = 3$. u_1 has 3 items in W_i, and the critical item set is I_1 and I_2, which has a total value of $110 > v_1 + v_2 + v_3 - G(u_1, Top(k,1)) = 70$, and sum of $UL(I_1) + UL(I_2) = 4$. Other eligible critical item sets include $\{I_1, I_3\}$ and $\{I_1, I_2, I_3\}$. By sorting the item on $v_j/UL(I_j)$, we pick up the items in order $\{I_2, I_1, I_3\}$. The final critical item set is $K_i = \{I_1, I_2\}$.

4.2.2 Item Insertion

When an item insertion occurs, the system retrieves all candidate users with some pruning condition, and re-computes the T1U2 exchange to update the top-k recommendations.

After a new item I_j is inserted into the wish list W_i of an user u_i, some new eligible exchange pairs are generated. If there is a new eligible exchange between user u_l and u_i, u_l must own this item in its unneeded item list L_i. Otherwise, this exchange pair must be tested before. Hence the candidate user set CU is initialized

with the inverted list $UL(I_j)$. Then for each user u_l in CU, the system examines if u_i owns a critical item of u_l or u_l owns a critical item of u_i. If any of these two cases happens, Algorithm 6 is invoked to find the optimal exchange pair between u_i and u_l.

We give an additional example of item insertion. In the example illustrated in Table 4.2, if one new item I_1 is inserted into u_2's wish list W_2, the system first retrieves the users owning I_1 in their unneeded item lists. Such users include u_3 and u_5. The system then tests if these candidate users have at least one critical item of u_2. Since u_5 does not contain any u_2's critical items $\{I_6\}$, and u_2 does not contain any u_5's critical items $\{I_4, I_6\}$ in the unneeded item list. Therefore, u_5 fails the test and u_3 will be further checked by the 2-user item exchange algorithm.

4.2.3 Item Deletion

When removing some I_j from W_i, the deletion operation can be done in two steps. In the first step, the system deletes all the current top-k exchanges containing the deleted item. In the second step, some re-computation is run to find new top-k exchange pairs for users with insufficient exchange recommendations.

The first step in the deletion operation is implemented with some inverted list structure, allowing the system to quickly locate all top-k exchange pairs with the deleted item I_j in W_i. Assume that the users with deleted exchange pairs are all kept in a "to-be-updated" user list. Algorithm 7 is then called for each user in the list, to fix all the top-k recommendation pairs. This implies that the deletion operation is expensive if many users are added into the "to-be-updated" user list.

To optimize the system performance, we propose some optimization technique possibly reducing the number of users in the "to-be-updated" user list after the deletion operation. The basic idea of the optimization is maintaining top κ ex-

change pairs for each user u_i, with some integer $\kappa > k$. It is straightforward to verify that $Top(k, i)$ is subset of $Top(\kappa, i)$. To utilize the expanded top exchange recommendation set, the system updates $Top(\kappa, i)$ for each insertion operation. On item deletion, if one of the exchange pair $E \in Top(\kappa, l)$ is removed due to the deletion of $I_j \in W_i$, the exchange list will not be totally re-computed immediately. Instead, the new T1U2 exchange between u_i and u_l is evaluated. If the new optimal exchange on u_i and u_l remains in $Top(\kappa, l)$, it is directly inserted back into $Top(\kappa, l)$. Otherwise, the counter decreases by one from κ to $\kappa - 1$. The complete re-computation of $Top(\kappa, l)$ is delayed until the next insertion operation on lists of u_l or there is less than k exchange pairs left with the system. We can prove that the all exchange pairs in $Top(k, i)$ must be exactly maintained by the scheme. Although it incurs more cost on insertions (because of the larger critical item set), this optimization greatly improves the overall performance of the system by cutting unnecessary re-computation of top exchange pairs.

We give an additional example of item deletion. Assume that $k = 2$ and $\kappa = 3$. At first, one user u_1 has 3 top exchanges: $E_1 = (u_1, u_3, \{I_1, I_2\}, \{I_5\})$, $E_2 = (u_1, u_5, \{I_1\}, \{I_4, I_6\}$ and $E_3 = (u_1, u_2, \{I_3\}, \{I_6\})$. If I_4 is deleted from L_1, E_2 is removed from the list, and κ_1 become 2. Suppose then I_6 is deleted, E_3 is also removed and κ_1 become 1. Then re-computing is triggered, and κ_1 is reset to 3, with the top results list re-computed.

4.3 Summary

In this chapter, we first designed an FPTAS to solve the two-user exchange algorithm. This approximation algorithm successfully solve the NP-hard problem in polynomial time with controllable approximation ratio. Then we propose a effi-

cient solution to the general Top-k Exchange Pair Monitoring problem, based on the critical item selection method.

CHAPTER 5

EXPERIMENT STUDY

In this chapter, we evaluate the algorithms we proposed in Chapter 4. We adapt the real life data from B2B online market as well as generating synthetic data based on some general models.

5.1 Data Generation and Experiment Settings

5.1.1 Synthetic Dataset

The first step of synthetic data generation is creating certain number of items. Each item is assigned with a value. Values are generated according to certain distributions, including exponential and Zipf distributions. The parameters of all the distributions in being investigated are provided in Table 5.1. The maximum value and minimum value are set at 10,000 and 10 respectively. When generating the item values, the distributions are truncated to keep all prices between 10 and 10,000.

In real system, users and their items are usually strongly correlated, because of the similar tastes and behaviors. To capture the diversity and clustering properties

Distribution	Density Function $p(x)$	Parameter
Exponential	$\lambda e^{-\lambda x}$	$\lambda = 1$
Zipf	$\frac{1/x^s}{\sum_{n=1}^{N}(1/n^s)}$	$s = 1$, $N = V_{max}$

Table 5.1: Parameters controlling the distributions on values

on the users and items, we setup 5 classes to model different types of users and their preferred items. Each user is randomly assigned to one of the classes with equal probability. One of the class is considered as "background class", which contains all the items. Every item is also assigned to one of the other four classes with equal probability. There is an upper limit on the maximum number of items in each list N. An item list, e.g. wish list W_i or unneeded item list L_i, is full if the number of items reaches the limitation. In our experiments, to test the scalability of the system, we try to keep the item list as full as possible.

After setting the parameters and assigning users and items to the classes, the synthetic data are generated with a sequence of item updates. The generation of updates consists of two phases. The first phase is the warm-up phase. The objective of this phase is to fill each user's wish and unneeded item lists, thereby with more insertions than deletions. After the lists are almost full, the second phase of simulation is started. In this phase, insertions and deletions take place with identical frequency, leading to relatively stable system workload.

In the first phase, when generating a new update, our simulation randomly selects a user with equal probability. The generator then chooses one of the wish list or the unneeded item list. If the target list is not full, an insertion operation is taken. Otherwise, the generator randomly deletes one of the item in the target list. During insertion, the selection on the inserting item depends on the user's class as well as the items' class. The generator picks up a random number to decide if the item is from the same class of the user (4/7 probability), the "background"

class (2/7 probability) or the other three classes (1/7 probability, and 1/21 for each class). It then uniformly chooses an item from the specific class. During deletion, one item is chosen from the list with equal probability. The selection of the deleting item does not take class information into account.

In the second phase, similar to the first phase, one item list from the chosen user is selected with equal probability. If the selected item list is empty, an insertion to the item list is done. If the item list is neither full nor empty, the generator makes a random decision: it generates an insertion with probability 0.6, or a deletion with probability 0.4. The selected probabilities are able to keep all lists almost full in the second phase.

The number of updates generated in the first phase is $N * |U|$, where $|U|$ is the number of users and N is the maximal number of items in any list. The number of updates generated in the second phase is no less than $2 * N * |U|$. The performance tends to turn stable after a series of updates in the second phase.

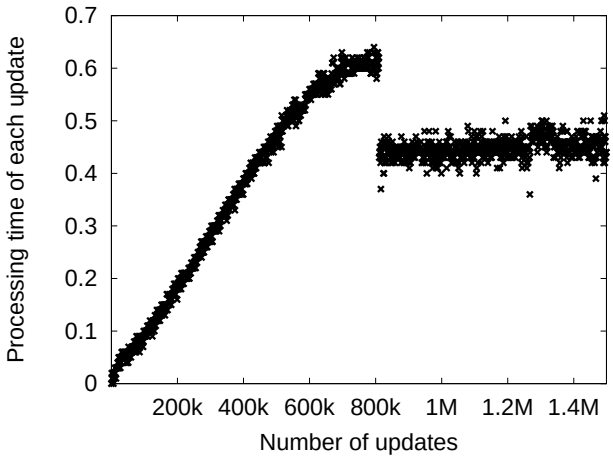
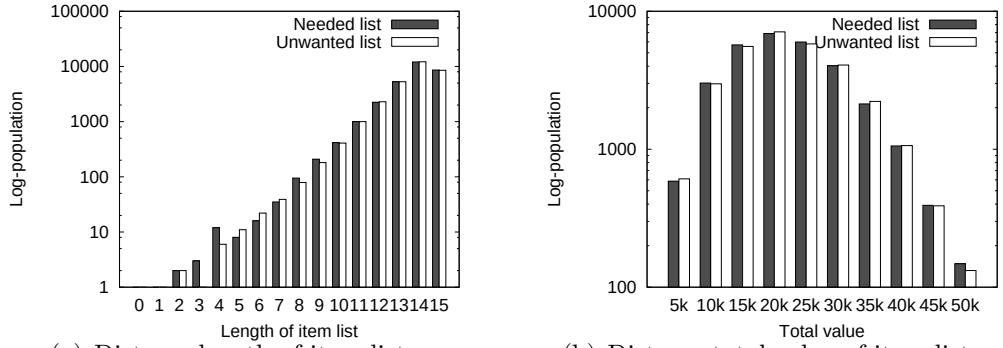


Figure 5.1: Average update response time over time

In Figure 5.1, we present the change in average update response time during our simulation. In the first phase of the simulation, the response time increases quickly. After transiting to the second phase, the performance tends to be stable.



(a) Dist. on length of item lists

(b) Dist. on total value of item lists

(c) Dist. on length of item list intersections

(d) Dist. on total value of item list intersections

Figure 5.2: Distribution on length and total value of user item lists and intersections

All our experimental results are collected in the second phase of the simulation.

The Figure 5.2 illustrates the distribution of the item after a period of running and the system performance has been stabilized. The number of users in the system is 30,000 and the length of item list is limited to 15. Figure 5.2(a) represents the distribution of item length of each user. As we can see in the figure, the majority of users have an item list that is almost full. More than 80% users' item lists are of length 13, 14 or 15. Figure 5.2(b) illustrates the distribution on total value of each user's item list. As shown in the figure, the total value is concentrated around 15k˜20k. Figure 5.2(c) shows the distribution on the length of the item list intersections, which is the number of common items between two users. It can be seen that users tend to have very small number of intersections. In most of the

cases, it is no more than 5 items. The same trend can be seen in Figure 5.2(d), which plots the distribution of intersection value between users. Among all $|U|^2$ pairs of users, only a several hundred user pairs share items with more than 20k total value.

Table 5.2 summarizes the parameters tested in our experiments. Their default values are in bold font.

Parameter	Varying Range
Number of users	$10k, 20k, \mathbf{30k}, 40k, 50k$
β	$0.7, 0.75, \mathbf{0.8}, 0.85, 0.9, 0.95$
Length of item list	$10, 15, \mathbf{20}, 25, 30$
κ	$15, 25, \mathbf{35}, 45, 55, 65, 75$
k	$1, 3, \mathbf{5}, 7, 9, 11$
Number of items	$300, 600, 900, 1200, \mathbf{1500}$
ϵ	$1 - \beta$

Table 5.2: Varying parameters in synthetic data set

5.1.2 Real Dataset

It is difficult to find real exchanging data from large online communities. To get a better understanding on our method with real world applications, we crawl some transaction data from eBay.com, which is a famous C2C online market system.

Our crawler records historical transactions with certain users in consecutive 90 days. Afterwards, all the users participating in these transactions are crawled in the same manner. In total we have crawled 34,191 users, 452,774 item records and 1,094,152 transaction records. We associate a user's wish (unneeded) list with all the item that he/she buys (sells).

As an online market is different from an exchanging market, we pre-process the data in order to make it suitable to test our system. We find that there are large number of duplicated or highly similar items. In order to reduce the duplication and increase the overlapping between user item lists, highly similar items are merged together. Some items and users are discarded to make sure that every user has non-empty item list. After the pre-processing, the final result data contains 2,458 users and 2,769 items.

To test our system performance under different number of users, we re-scale the data to generate data set of various size. To scale up the data, we randomly duplicate existing users until reaching the desired size. The duplicated user associates with the same set of items. To scale down the data, we randomly remove users.

We generate continuous updates according to the transactions we have crawled. We associate an item with a user's wish (unneeded) list, if this user have bought (sold) this item. To generate update operations, we randomly choose a user, an updating type (insertion/deltetion), an item list (wish/unneeded) and an item associated with this list.

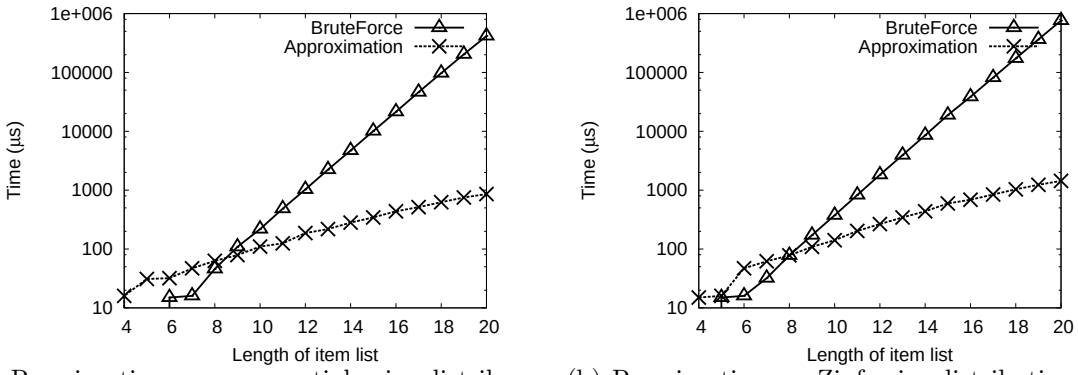
The length of an item list at any moment is limited within 15. A list with 15 items are considered as full. The reason to set a fixed limitation is that our crawled transactions span 90 days. These items are not listed at the same time. At any moment, only a small number of items are listed. Therefore, we set this fixed limitation to control the number of items simultaneously listed in an item list.

Table 5.3 summarizes the parameters tested in our real data experiments. Their default values are in bold font.

Parameter	Varying Range
Number of users	$0.5k, 1.5k, \mathbf{2.5k}, 3.5k, 4.5k$
β	$0.7, 0.75, \mathbf{0.8}, 0.85, 0.9, 0.95$
κ	$15, 25, \mathbf{35}, 45, 55, 65, 75$
k	$1, 3, \mathbf{5}, 7, 9, 11$
ϵ	$1 - \beta$

Table 5.3: Varying parameters in real data set

5.2 Experiments on T1U2 Exchange



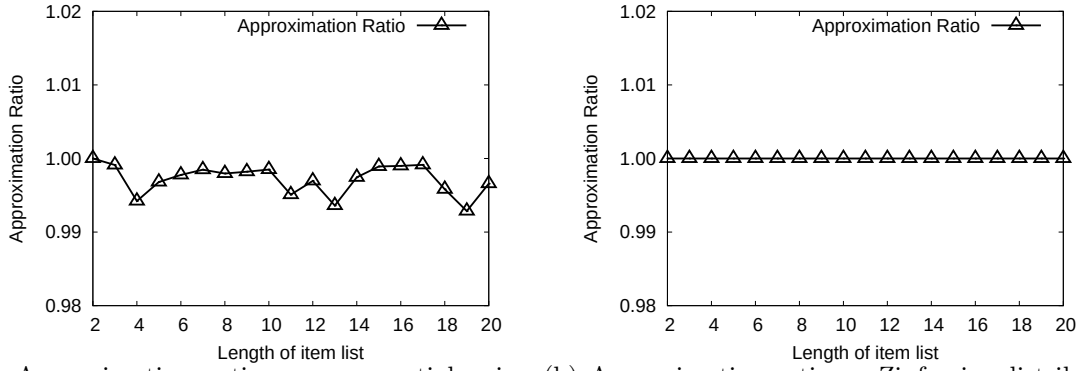
(a) Running time on exponential price distribution

(b) Running time on Zipf price distribution

Figure 5.3: Impact of varying item list length on running time

In Section 4.1 we propose Algorithm 6, which is an approximation algorithm for finding T2U1 exchange. In this section, we evaluate its performance, including the running time and the approximation ratio. Also we use the brute force algorithm as straw-man. We test both algorithms on exponential and Zipf distribution. Detailed density functions and parameters of them are as shown in 5.1.

Figure 5.3 and 5.4 present the performance of both algorithms under item lists of different length. We fix both β and $1 - \epsilon$ to 0.8, and generate two item lists



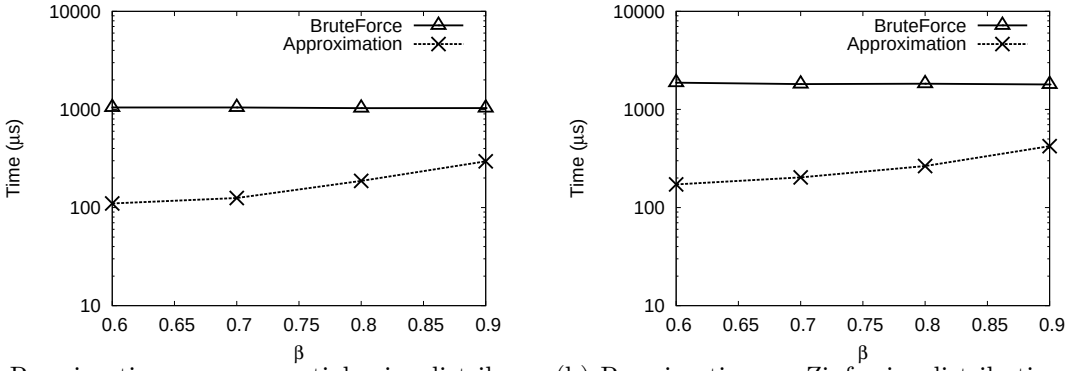
(a) Approximation ratio on exponential price distribution

(b) Approximation ratio on Zipf price distribution

Figure 5.4: Impact of varying item list length on approximation

of equal length, as $W_i \cap L_l$ and $L_i \cap W_l$. Figure 5.3 shows the running time of both algorithms. As the plots shown, when the lengths of the item lists are less than 8, approximation scheme is not as good as brute-force algorithm, because approximation method spends too much time on index construction. However, as the size of the item set grows, the running time of brute force algorithm grows exponentially, while the approximate algorithm shows a good scalability. Figure 5.4 represents the approximation ratio of the approximate T1U2 algorithm on various value distributions. The approximation ratio is defined as the proportion of the approximated result to the accurate result, i.e. the output of the brute force algorithm. The results show that under either value distribution, the approximation ratio is no smaller than 0.99.

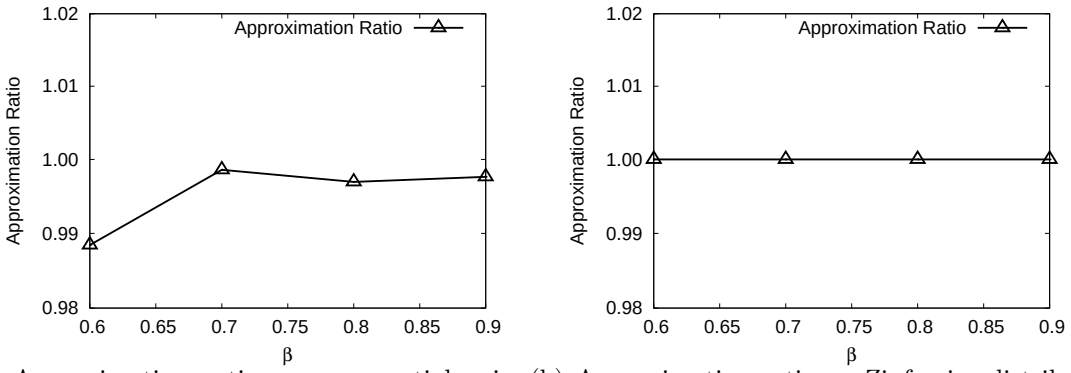
Figure 5.5 discusses the effect of relaxation ratio β on the running time of both algorithms, when the number of items are fixed at 10. We set ϵ for Algorithm 6 at $1 - \beta$. The running time of Algorithm 6 increase with β, which well follows the complexity analysis. On the other hand, β does not affect the running time of brute-force method. Figure 5.6 shows that the actual approximation ratio in practice is much better than the theoretical estimation.



(a) Running time on exponential price distribution

(b) Running time on Zipf price distribution

Figure 5.5: Impact of varying β on running time



(a) Approximation ratio on exponential price distribution

(b) Approximation ratio on Zipf price distribution

Figure 5.6: Impact of varying β on approximate rate

5.3 Top-K Monitoring on Synthetic Dataset

We compare our proposed algorithm with critical item pruning, referred to as 'Critical', with a basic algorithm, referred to as 'Basic'. The basic algorithm is similar to our proposed method. It finds the exchange candidates with the inverted list. However, it does not apply critical item pruning strategy. After exchange candidates are found, the algorithm simply find eligible exchange pairs between current user and each candidate using the T1U2 algorithm.

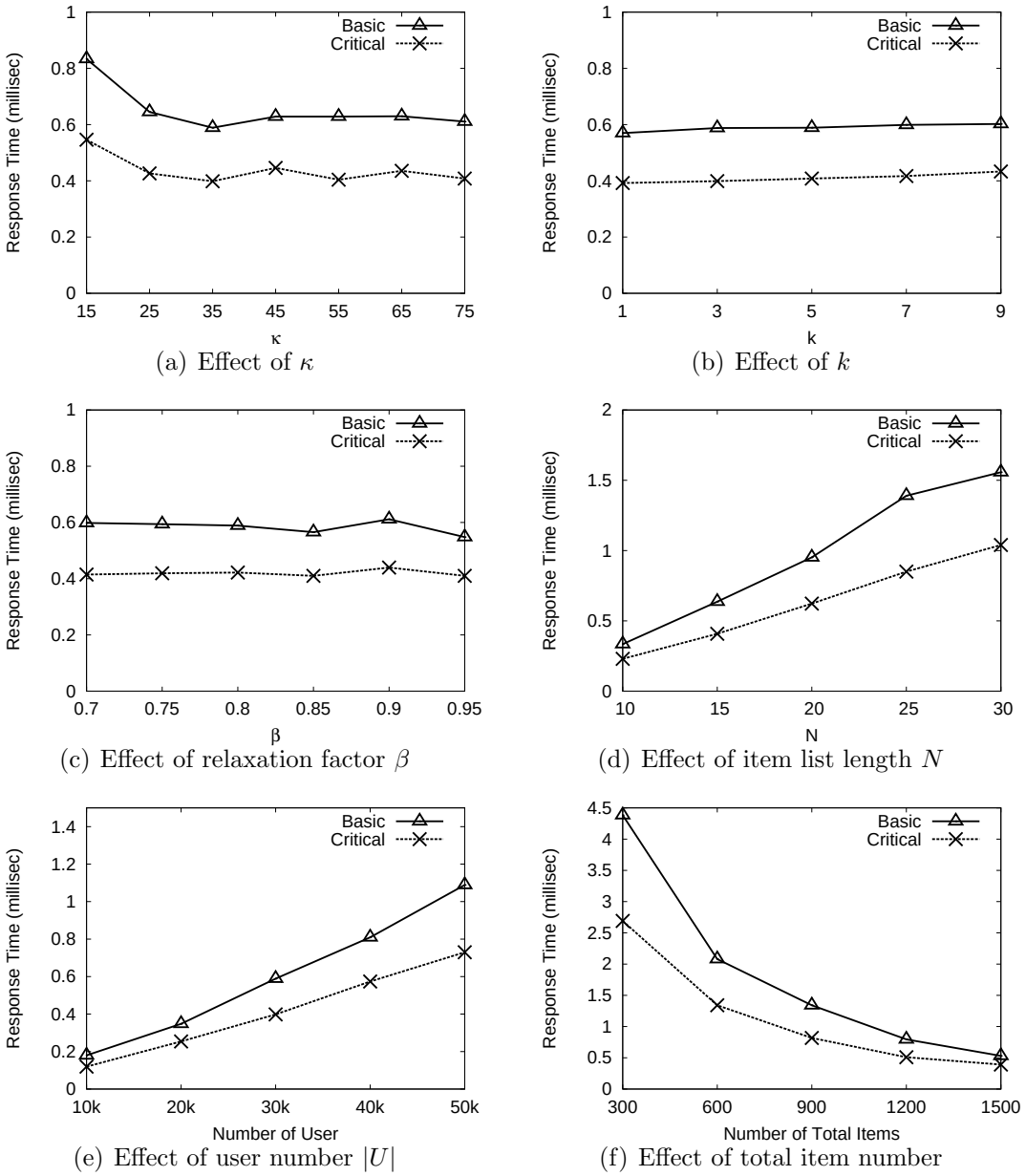
To verify the efficiency, we measure the response time. Only the experiment

results on exponential distribution are summarized, because there is no significant differences among results on various distributions. For each set of experiments, a query file is generated according to the rule we describe in Section 5.1. The query file contains 10 to 30 million updates and is long enough to makes sure that the system finally levels off. The average response time is measured every 1,000 continuous operations. The aim of our experiments is to test the impact of system parameters, the item price distributions and the user number.

As mentioned in Section 4.2.3, to optimize the performance, the system initially computes the top κ results instead of k, where $\kappa > k$. When one of the old top-k exchanges is deleted, top-κ results are calculated instead of re-computing only top-k results. We first test the impact of the number κ. The empirical result is also used to justify our selection of the default value for κ in Table 5.2.

The selection of κ affects the system performance on two sides. On the one hand, large κ decreases the frequency of re-computing. On the other hand, it increases the update cost. Figure 5.7(a) illustrates the system response time when varying κ, when k is set as default value 5. The result shows that the response time reduces when κ increases. The optimal performance is achieved when $\kappa = 35$ for both algorithms. When κ keeps increasing, the system performance levels off, because of the increasing cost of updates.

Then we study the effect of k, i.e. the number of top exchange recommendations. We record the system response time under different values of k. Figure 5.7(b) shows that the overall response time slightly increases with the growth on k. However, this minor increase makes no significant impact on the overall performance. This implies that the extra overhead brought by increasing k is not an important factor for our system. For basic algorithm, it scans the list and finds the candidate user. Therefore, its running time does not depend on k. For critical algorithm,



(a) Effect of κ (b) Effect of k

(c) Effect of relaxation factor β (d) Effect of item list length N

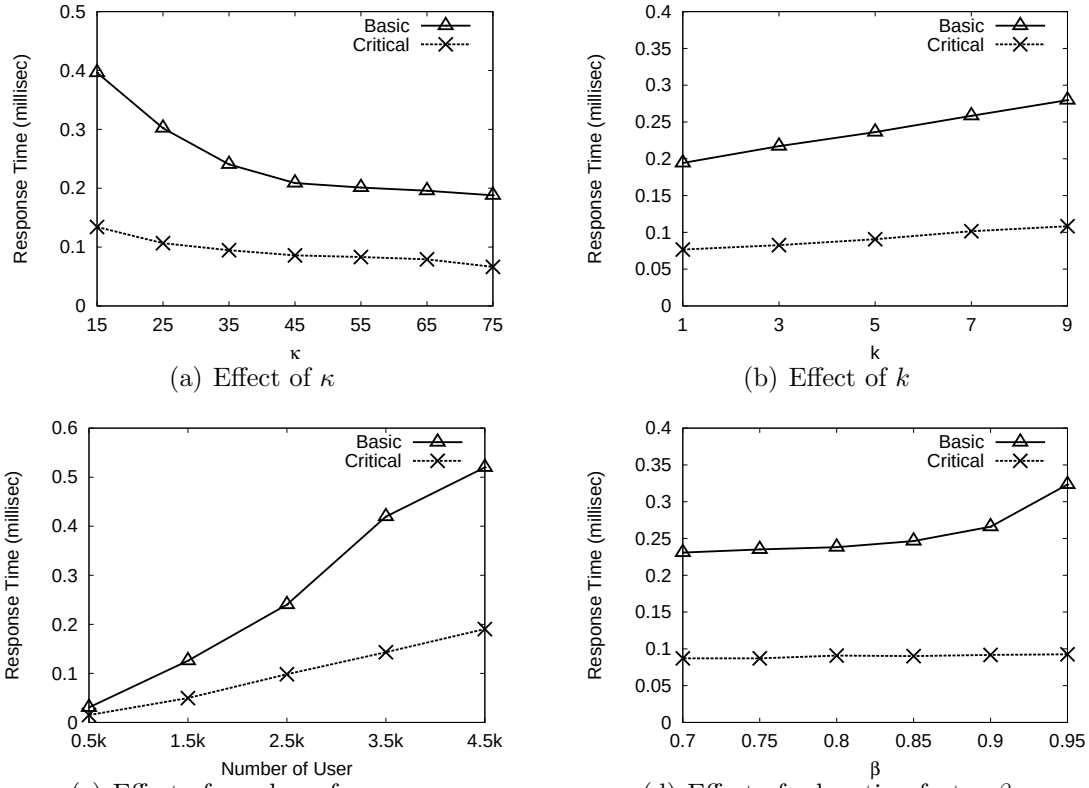
(e) Effect of user number $|U|$ (f) Effect of total item number

Figure 5.7: Top-K monitoring results on synthetic dataset

although increasing k can result in a larger critical item set, the pruning result is not significantly increased. This suggests that our pruning method is effective in reducing the candidate set size.

We next study the effect of relaxation factor β on the system performance. We illustrate the response time under different β factor, as shown in Figure 5.7(c). The

overall performance always holds on a certain level. This result implies that our system can work well under different β values. Response time of basic algorithm at $\beta = 0.95$ slightly decline in both data sets, since fewer eligible exchange can be found when the relaxation rate is higher.



(a) Effect of κ (b) Effect of k

(c) Effect of number of user u (d) Effect of relaxation factor β

Figure 5.8: Top-K Monitoring Results on Real Life Dataset

In our experiments, each user's item list is length fixed. It challenges the system performance when each user is allowed to list more items. We hereby study the performance on different lengthes of item lists. As shown in Figure 5.7(d), when the item list grows larger, the response time grows linearly with N. When the item list expands, items are more likely to appear in lists for different users. The system has to examine more users to update the exchange recommendations. In practice, users in online communities does not have a long item list. Therefore, the

current performance of our system is capable of handling the workload of general community systems.

Number of users in the system is another very important factor which greatly impacts the system performance. We evaluate the response time under different number of users. The result is presented in Figure 5.7(e). The result shows that the response time linearly grows with the number of users. Despite the decline of the system throughput, the performance of our method is still excellent even for the largest u we have tested (more than 1,000 updates per second under 50,000 users).

According to our data generating method, when the number of total items decreases, every item is shared by more users. This brings extra overhead to the system. It is reflected in our test of the system performance with varying number of items. As shown in Figure 5.7(f), the system performance is inversely proportional to the number of items.

5.4 Top-K Monitoring on Real Dataset

Similarly to the experiments in previous subsection, we compare "Critical" against "Basic" on real dataset. Firstly, we study the effect of κ, which is the initial top results that the system computes. In the tests, k is set at 5. The result is illustrated in Figure 5.8(a). As can be seen in the figure, response time keeps decreasing with κ increases. For the Basic algorithm, the response time drops significantly before $\kappa = 45$ and levels off after the point. The critical pruning algorithm is not greatly affected by the κ. Its response time decrease insignificantly with κ increases.

Secondly, we study the effect of k, which is the number of top results requested by user. The result is illustrated as Figure 5.8(b).

The result implies that our pruning strategy can well handle the increasing number of k. For both algorithms, the response time linearly increases with k. The critical algorithm increases slightly slower than the basic algorithm. The overall efficiency shows that our pruning strategy halves the response time. The improvement is better, because in a real life data set, item price distribution is more skewed and user-item ownership are more clustered.

Thirdly, we study the effect of u, which is the number of users participating in the exchange. We test both algorithm under various number of users. As our original (filtered) data set contains 2,458 users, we re-scale the data to generate differently sized data set. We down-scale the data set to generate $u = 500$ and $u = 1,500$ data sets. We up-scale the date to generate $u = 2,500$, $3,500$ and $4,500$ data sets. The result is shown in Figure 5.8(c).

The result shows that the critical algorithm has a high efficiency and nice scalability. It has an improvement up to near three times. When the user number increases, the response time of critical algorithm grows in a linear manner. Meanwhile, response time of basic algorithm grows faster when user number exceed 2,500. This is because that on the one hand, when we up-scale the data, each item is owned by more user, and the cost of searching for top-k exchange becomes more expensive; on the other hand, each deleting effects more top-k results, which result in a more frequent top-k re-computing. As a result, the basic algorithm shows a super-linear increasing. Since the critical algorithm is less affected by re-computing frequency, it shows a linear growth in response time.

Lastly, we study the effect of β, which is the relaxation factor and also the approximation factor in Algorithm 6. The result is illustrated as Figure 5.8(b). The critical algorithm perform well under all β, while the response time of the basic algorithm keeps on increasing with β. In a real-life data, user-item ownership

are highly clustered. Therefore, small user group often shares a long common item list. In this case, the approximate T1U2 algorithm is launched more frequently than in our synthetic data set. As the approximation algorithm has an time complexity related to $(1-\beta)^{-1}$, the response time increase with β.

5.5 Summary

In this chapter, we empirically study our solution. We first evaluate our approximation T1U2 algorithm with synthetic data. We compare our proposed algorithm with brute-force under various item list length and β. The results shows our approximation algorithm can easily handle long item list without running time's explosive growing. The actual approximation ratio is also much better than the theoretical expectation.

We then study our general Top-k Recommendation Monitoring algorithm. Both synthetic and real-life data set are used. We compare our algorithm with a basic algorithm in which critical item selection is not used. We evaluate the impact of κ, k, β, N, $|U|$ and total item numbers on both algorithms. In all experiments, our our algorithm over perform the basic algorithm. Moreover, the experiment results show that our algorithm has a good scalability in terms of item list length, user number and total item number.

CHAPTER 6

CONCLUSION

In this thesis, we study the problem of top-k exchange pair monitoring on large online community system. We propose a new exchange model, Binary Value-based Exchange Model (BVEM), which allows exchange transaction between users only when they both have items the other side wants and the total values of the items are of the same price. We present an efficient mechanism to find the top-1 exchange pair between two users, and extend the analysis to large system with arbitrarily many users. Extensive experiments on synthetic data sets show that our solution provides a scalable and effective solution to the problem.

As a future work, we are planning to extend our model by adding or relaxing constraints in Definition 3.1. For example, the condition on exact item match can be replaced by type match, allowing user to claim general type of item in his/her wish list. Spatial constraint, as another example, can help the users to find the exchange opportunities more convenient to proceed. It is also interesting to investigate the possibilities of new exchange models in the social networks, utilizing the relationships among users.

BIBLIOGRAPHY

[1] http://gamersunite.coolchaser.com/games/frontierville.

[2] http://singapore.gumtree.sg/.

[3] http://transplants.ucla.edu/body.cfm?id=112.

[4] http://www.dcs.gla.ac.uk/ pbiro/applications/uk_ke.html.

[5] http://www.iswap.co.uk/home/home.asp.

[6] http://www.shede.com.

[7] Zeinab Abbassi and Laks V. S. Lakshmanan. On efficient recommendations for online exchange markets. In *ICDE*, pages 712–723, 2009.

[8] Zeinab Abbassi and Laks V. S. Lakshmanan. On efficient recommendations for online exchange markets. In *ICDE*, pages 712–723, 2009.

[9] Atila Abdulkadiroglu and Tayfun Sonmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–702, May 1998.

[10] David J. Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. In *ACM Conference on Electronic Commerce*, pages 295–304, 2007.

[11] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[12] Asim Ansari, Skander Essegaier, and Rajeev Kohli. Internet Recommendation Systems. *Journal of Marketing Research*, 37(3):363–375, August 2000.

[13] J. S. Armstrong. *Principles of Forecasting - A Handbook for Researchers and Practitioners (International Series in Operations Research & Management Science)*. Springer, 2001.

[14] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[15] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[16] Daniel Billsus and Michael J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3):147–180, February 2000.

[17] Péter Biró and Katarína Cechlárová. Inapproximability of the kidney exchange problem. *Information Processing Letters*, 101(5):199 – 202, 2007.

[18] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth*

conference on Uncertainty in artificial intelligence, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[19] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[20] Katarína Cechlárová and Vladimír Lacko. The kidney exchange problem: How hard is it to find a donor? *Annals OR*, 193(1):255–271, 2012.

[21] Yueguo Chen, Su Chen, Yu Gu, Mei Hui, Feng Li, Chen Liu, Liangxu Liu, Beng Chin Ooi, Xiaoyan Yang, Dongxiang Zhang, and Yuan Zhou. Marcopolo: a community system for sharing and integrating travel information on maps. In *EDBT*, pages 1148–1151, 2009.

[22] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, California, 1999. ACM.

[23] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[24] Souvik Debnath, Niloy Ganguly, and Pabitra Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 1041–1042, New York, NY, USA, 2008. ACM.

[25] Joaquin Delgado and Naohiro Ishii. Memory-based weighted-majority prediction for recommender systems. In *ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, August 1999.

[26] F. L. Delmonico. Exchanging kidneys–advances in living-donor transplantation. *N. Engl. J. Med.*, 350(18):1812–1814, Apr 2004.

[27] Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political Economy*, 87(2):293–314, April 1979.

[28] Hideo Konishi, Thomas Quint, and Jun Wako. On the shapleycscarf economy: the case of multiple types of indivisible goods. *Journal of Mathematical Economics*, 35(1):1 – 15, 2001.

[29] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[30] Ken Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[31] M. Lucan, P. Rotariu, D. Neculoiu, and G. Iacob. Kidney exchange program: a viable alternative in countries with low rate of cadaver harvesting. *Transplant. Proc.*, 35(3):933–934, May 2003.

[32] Jinpeng Ma. Strategy-proofness and the strict core in a market with indivisibilities. *International Journal of Game Theory*, 23:75–83, 1994. 10.1007/BF01242849.

[33] Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[34] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM confer-*

ence on Digital libraries, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.

[35] Atsuyoshi Nakamura and Naoki Abe. Collaborative filtering using weighted majority prediction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 395–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[36] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification ofinteresting web sites. *Mach. Learn.*, 27(3):313–331, Junaury 1997.

[37] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev.*, 13(5-6):393–408, December 1999.

[38] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Collaborative Work Conference*, pages 175–186, Chapel Hill, NC, October 1994. Association of Computing Machinery, Association of Computing Machinery.

[39] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3(4):329 – 354, 1979.

[40] Alvin E. Roth. Incentive compatibility in a market with indivisible goods. *Economics Letters*, 9(2):127 – 132, 1982.

[41] Alvin E. Roth and Andrew Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4(2):131–137, August 1977.

[42] Alvin E. Roth and Tayfun S˙ Pairwise kidney exchange. *Journal of Economic Theory*, 125(2):151 – 188, 2005.

[43] Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Kidney exchange. *The Quarterly Journal of Economics*, 119(2):457–488, 2004.

[44] Alvin E. Roth, Tayfun Oguz Sonmez, and M. Utku Ünver. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, 97(3), 2007.

[45] Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[46] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989.

[47] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[48] Mark A Satterthwaite and Hugo Sonnenschein. Strategy-proof allocation mechanisms at differentiable points. *Review of Economic Studies*, 48(4):587–97, October 1981.

[49] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, March 1974.

[50] Tayfun Sönmez and M. Utku Ünver. Market design for kidney exchange. Technical report. forthcoming.

[51] Tayfun Sönmez and M. Utku Ünver. Matching, allocation, and exchange of discrete resources. Boston College Working Papers in Economics 717, Boston College Department of Economics, August 2009.

[52] Vijay V. Vazirani. *Approximate Algorithms*. Springer, 2003.